НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
Олександр ПАВЛОВ
(підпис) (вл.ім'я, прізвище)

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «*Інформаційна система підтримки сервісу розповсюдження фотознімків*»

Виконав: студент *IV* курсу, групи *ІС-62*
Сєвєрцев Вадим Вячеславович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст. вик. Ковтунець Олесь Володимирович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц. Новінський Валерій Петрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *к.т.н., доц. Пасько Віктор Петрович*
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
Олександр ПАВЛОВ
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

ЗАВДА ННЯ
на дипломний проєкт студенту

Сєвєрцеву Вадиму Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту «*Інформаційна система підтримки сервісу розповсюдження фотознімків»*
керівник проєкту *Ковтунець Олесь Володимирович, ст. вик.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "*7*"*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту "*01*"*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення

5. *Технологічний розділ: керівництво користувача, методика випробувань програмного продукту*

5. Перелік графічного матеріалу

1. *Схема структурна варіантів використань*
2. *Схема структурна послідовності*
3. *Схема бази даних*
4. *Схема структурна класів*
5. *Схема структурна діяльності*
6. *Рішення з математичного забезпечення*

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*13.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*19.04.2020*	
3.	*Постановка та формалізація задачі*	*22.04.2020*	
4.	*Розробка інформаційного забезпечення*	*24.04.2020*	
5.	*Алгоритмізація задачі*	*25.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*29.04.2020*	
7.	*Розробка програмного забезпечення*	*12.05.2020*	
8.	*Налагодження програми*	*25.05.2020*	
9.	*Виконання графічних документів*	*20.05.2020*	
10.	*Оформлення пояснювальної записки*	*21.05.2020*	
11.	*Подання ДП на попередній захист*	*13.05.2020*	
12.	*Подання ДП на основний захист*	*06.06.2020*	
13.	*Подання ДП рецензенту*	*08.06.2020*	

Студент Вадим Сєвєрцев

Керівник Олесь Ковтунець

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6221.00.000 ПЗ	Пояснювальна записка	86	
3	А4	ДП 6221.01.000 ТЗ	Технічне завдання	10	
4	А3	ДП 6221.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6221.03.000 ССС	Схема структурна послідовності	3	
6	А3	ДП 6221.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6221.05.000 ССК	Схема структурна класів	1	
8	А3	ДП 6221.05.000 ССД	Схема структурна діяльності	4	
9	А3		Рішення з математичного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6221 00.000.00		
Розробн.		Сєверцев В.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62	
Н/контр.		Новінський В.П.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: «Інформаційна система підтримки сервісурозповсюдження фотознімків»

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’ятьох розділів, містить 39 таблиць, 11 рисунків, 1 додаток, 8 джерел.

Дипломний проект присвячений створенню інформаційної системи підтримки сервісу розповсюдження фотознімків. Розглянуті базові принципи побудови, розробки та тестування системи.

У розділі «Загальні положення» описується предметне середовище дипломного проекту, наводяться цілі дипломного проекту та задачі, які необхідно вирішити в процесі розробки. Описано функціональну модель, процеси діяльності та наведено функціональні вимоги до програмного продукту.

У розділі «Інформаційне забезпечення» наведено структуру бази даних та описано сутності, які фігурують в процесах системи.

У розділі «Програмне та технічне забезпечення» наводяться та описуються засоби розробки, які були використанні для розробки системи. Наводяться вимоги до технічного забезпечення.

У розділі «Технологічний розділ» описуються цілі та методи тестування програмного забезпечення. У цьому ж розділі наведено керівництво користувача.

ЗОБРАЖЕННЯ, ВЕБ-СЕРВІС, РЕКОМЕНДАЦІЇ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, WEB API

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of five sections, contains 39 tables, 11 drawings, 1 supplement, 8 sources.

The diploma project is dedicated to the creation of an information support system for the photo distribution service. The basic principles of building, developing and testing the system are considered.

The section "General provisions" describes the subject environment of the diploma project, provides the goals of the diploma project and the tasks that need to be solved in the development process. The functional model, activity processes, and functional requirements for the software product are described.

The section "information support" shows the structure of the database and describes the entities that appear in the system processes.

The section "Software and hardware" provides and describes the development tools that were used to develop the system. Requirements for technical support are given.

The "Technology section" describes the goals and methods of software testing. The user's guide is also provided in this section.

IMAGES, WEB SERVICE, RECOMMENDATIONS, COLLABORATIVE FILTERING, INFORMATION SYSTEM, DATABASE, WEB API

ЗМІСТ

ВСТУП

Разом з розвитком інформаційних систем, Інтернет-технологій та глобальною інформатизацією суспільства також збільшується інформаційне навантаження на користувачів інтернет ресурсів. В наш час, зі збільшенням кількості інформації, що є доступною у мережі Інтернет, процес пошуку необхідної інформації включає в себе перегляд великої кількості інтернет джерел та ресурсів.

Сьогодні зображення та фотознімки є популярним джерелами інформації серед користувачів мережі Інтернет, а сервіси для обміну зображеннями та їх розповсюдження знаходяться в тренді.

Одними з найголовніших завдань для будь-яких інформаційних ресурсів є обробка та аналіз даних. У наш час використання інформаційних систем значно спрощує процеси збору, аналізу, обробки та використання інформації. Зазвичай роль інформаційних систем виконує спеціалізоване програмне забезпечення.

Основними цілями розробки інформаційної системи підтримки сервісу для розповсюдження фотознімків є забезпечення механізмів збору, збереження, аналізу даних сервісу та прискорення процесів пошуку та розповсюдження фотознімків між користувачами сервісу.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Збереження, обробка та аналіз даних є невід'ємною частиною функціонування будь-якого сервісу, що дозволяє користувачам переглядати та ділитися контентом того чи іншого роду. Фільтрація контенту, забезпечення якості, актуальності та доступності контенту напряму впливають на популярність сервісу та дозволяють забезпечити інформаційні потреби користувачів. Інформаційні системи дозволяють організувати та налаштувати ці процеси.

Інформаційна система — це комунікаційна система, що реалізує функції збору, зберігання, обробки і передачі інформації. Найважливішими функціями такої системи є прогнозування, планування, облік, аналіз, контроль і регулювання.

Процеси, що забезпечують роботу інформаційної системи складаються з таких основних етапів:

- збір інформації із зовнішніх або внутрішніх джерел, корегування та фільтрування інформації;
- обробка інформації та її структуризація;
- перетворення інформації для її передачі споживачам або для подальшого аналізу у системі.

Інформаційна система визначається наступними властивостями:

– будь-яка інформаційна система може бути побудована й керована на основі загальних принципів побудови систем;

– інформаційна система є динамічною й такою, що розвивається;

– при побудові інформаційної системи необхідно використати системний підхід;

– вихідною продукцією інформаційної системи є інформація, на основі якої приймаються рішення.

Основними процесами функціонування сервісу для розповсюдження фотознімків є зберігання фотознімків та зображень, їх пошук, перегляд та розповсюдження між користувачами сервісу.

Інформаційна система, що розробляється для підтримки сервісу розповсюдження фотознімків, забезпечуватиме механізми збереження зображень користувачів, пошуку та фільтрування контенту для користувачів відповідно до їх запитів, формування рекомендацій для користувачів відповідно до їх вподобань, основуючись на інформації щодо їх активності у сервісі.

1.1.1 Опис процесу діяльності

Процес діяльності починається з авторизації користувача у системі. Щоб авторизація була можливою, користувач повинен зареєструватися в системі. Після входу в систему користувач веб-сервісу може змінювати дані профілю, переглядати зображення серед колекцій сервісу та оцінювати їх, відмічаючи, що зображення їм сподобалось, виконувати пошук та фільтрування зображень за тегами, завантажувати власні зображення до колекцій сервісу.

Для реєстрації в сервісі користувач вводить свою електронну пошту, ім’я користувача, власне ім’я, власне прізвище, пароль та підтвердження паролю. Для пошуку та фільтрації зображень користувач вводить в поле для пошуку теги, яким повинні відповідати зображення. Для завантаження власних зображень до колекції сервісу користувач вводить заголовок для посту, набір тегів, через пробіл, які будуть прив’язані до зображення та обирає зображення, яке необхідно завантажити. Для зміни даних профілю

користувача, користувач вводить у поля для даних, які необхідно змінити, нові актуальні дані та зберігає зміни.

Діаграма діяльності загальна, діаграма діяльності пошуку зображень, діаграма діяльності авторизації користувача в системі та діаграма діяльності створення посту наведені у графічному матеріалі.

1.1.2 Опис функціональної моделі

Дійовою особою є користувач, який реєструється для користування сервісом та може здійснювати наступні дії: він має змогу передивлятися та оцінювати зображення колекції сервісу, виконувати пошук та фільтрації зображень колекції сервісу, завантажувати власні зображення до колекції сервісу. Також зображення, що відображатимуться на головній сторінці сервіс, будуть фільтруватися в залежності від активності користувача.

Таблиця 1.1. – Функціональні вимоги

Актор	**Варіант використання**	**Функціональна вимога**	**Пріоритет**
Користувач	Реєстрація в системі	1. Система надає користувачу можливість вводити дані для реєстрації 2. Система перевіряє коректність введених реєстраційних даних 2.1 Система повідомляє користувача про	Високий

Продовження таблиці 1.1

		помилку введених даних 2.2 Система повідомляє користувача про успішну реєстрацію у системі	
Користувач	Вхід до системи	3. Система надає користувачу можливість вводити дані для авторизації. 4. Система перевіряє на наявність авторизаційних даних користувача у системі 4.1 Система повідомляє користувача про помилку введених даних, якщо авторизаційних даних немає у системі 4.2 Система надає користувачу доступ до системи, якщо ввів авторизаційні дані, що наявні у системі	Високий

Продовження таблиці 1.1

Користувач	Перегляд зображень	5. Система надає користувачу можливість переглядати зображення	Високий
Користувач	Пошук та фільтрація зображень	6. Система надає користувачу можливість виконувати пошук та фільтрувати зображення по тегам	Високий
Користувач	Завантаження зображень до системи	7. Система надає користувачу можливість завантажувати його зображення до системи 7.1 Система надає користувачу можливість вводити заголовок для зображення 7.2 Система надає користувачу можливість вводити теги для зображення	Високий

Продовження таблиці 1.1

		7.3 Система надає користувачу можливість обирати зображення для завантаження	
Користувач	Оцінювання зображень	8. Система надає користувачу можливість відмічати зображення, які йому сподобалися	Середній
Користувач	Видалення власних зображень з колекції системи	9. Система надає користувачу можливість видаляти зображення, які він завантажував до системи, з колекції системи.	Середній
Користувач	Зміна даних профілю користувача	10. Система надає користувачу можливість змінювати дані профілю користувача 10.1 Система надає користувачу можливість змінювати	Середній

Продовження таблиці 1.1

		електронну пошту 10.1.1 Система перевіряє коректність введеної електронної пошти. 10.1.2 Система повідомляє користувача про помилку введення електронної пошти, якщо введена електронна пошта вже використовується або має не коректний формат 10.2 Система надає користувачу можливість змінювати інформацію про його ім’я користувача 10.3 Система надає користувачу можливість змінювати інформацію про його	

Продовження таблиці 1.1

		ім’я. 10.4 Система надає користувачу можливість змінювати інформацію про його прізвище	

Схема структурна варіантів використання наведена у графічному матеріалі.

1.2 Огляд наявних аналогів

Pinterest

Pinterest - це медіа соціальна мережа, яка дозволяє користувачам ділитися зображеннями, пов'язаними з роботою, природою, товарами, відпочинком, послугами та іншими складовими нашого життя, та візуально відкривати нові інтереси, переглядаючи зображення, які розмістили інші користувачі ресурсу. Користувачі мають змогу підписуватися об’єднуватися у групи за вподобаннями та ділитися тематичними знімками один з одним.

Unsplash

Unsplash - це веб-сайт, призначений для обміну фотографіями за ліцензією Unsplash. Веб-сайт заявляє про понад 110 000 фотографів, що надсилають участь, і створює понад 11 мільярдів переглядів фотографій на місяць у своїй зростаючій бібліотеці фотографій.

Unsplash названий одним із провідних веб-сайтів у галузі фотографії. Фотографи завантажують фотографії на веб-сайт, а команда редакторів Unsplash займається їх подальшим просуванням. Дозвільні умови авторських прав на його фото графії призвели до того, що Unsplash став одним з найбільших постачальників фотографій в Інтернеті, а фотографії користувачів часто з’являються у статтях.

We heart it

We Heart It – це візуальна платформа, яка підтримує зображення, анімовані GIF-файли та відео. Користувачі можуть зберігати (або «сердечно позначати») свої улюблені зображення, щоб поділитися ними з друзями та організувати їх у колекції.

Порівняння

Наведені вище системи дають користувачу змогу обрати теми, які їм цікаві. Далі ці теми використовуються системами для наповнення їх стрічки контенту зображеннями з заданою тематикою.

Розроблювана ж система дає користувачу змогу виконувати пошук зображень за тегами та буде наповнювати стрічку контенту зображеннями з шуканою тематикою, але також, в залежності від активності користувача, система буде шукати тізображення, що можуть сподобатися користувачу, але пошук яких він не виконував, та додавати їх у стрічку контенту користувача.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки є забезпечення механізмів збереження, пошуку, редагування та рекомендацій контенту для сервісу розповсюдження фотознімків.

1.3.2 Цілі та задачі розробки

Цілями розробки є:

- розробка сервісу розповсюдження фотознімків та інформаційної системи для його підтримки;

- забезпечення зручного способу обміну, розповсюдження та збереження зображень або фотознімків;

- забезпечення можливості оцінювати та шукати зображення та фотознімки за вподобаннями або потребами

- формування рекомендацій для користувачів сервісу відповідно доїх активності у сервісі.

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

1) Створити веб-сайт для сервісу;
2) Розробити інформаційну систему, що забезпечуватиме збір, обробку та доступ до даних сервісу:

- реєстрація користувачів;
- завантаження зображень та фотознімків до колекції сервісу;
- оцінювання зображень та фотознімків інших користувачів;
- пошук та фільтрація зображень;
- перегляд зображень з колекції сервісу.

3) Розробити та реалізувати алгоритм для рекомендації зображень користувачам, відповідно до їхньої активності у сервісі;

Висновок до розділу

Було вивчено основні бізнес-процеси, що виконуються в рамках використання сервісу для розповсюдження фотознімків. Були виділені успішні та відомі проекти, що стосуються даної предметної області. Визначено варіанти використання, функціональні вимоги.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Дані до системи надходять від однієї сутності – актора системи – користувача.

Дані від користувача надходять в наступних випадках:

- при реєстрації;
- при авторизації;
- при внесені змін до профілю користувача;
- при виконанні пошуку та фільтрації зображень серед колекції системи;
- при завантаженні зображень та фотознімків користувача до колекції системи.

2.2 Вихідні дані

Даними, що повертаються системою, є:

- список сутностей системи, що результатом виконання запиту користувача під час пошуку та фільтрування сутностей системи;
- список рекомендацій для користувача;
- відповідь системи щодо результату виконання запиту при збереженні, видаленні, оновленні даних.

2.3 Опис структури бази даних

У даному розділі представлено опис структури бази даних системи.

Для зберігання даних системи було вирішено використовувати реляційну базу даних. Середовище розробки – MS SQL.

У базі даних зберігається інформація про такі сутності системи:

- користувач;

- пост;

- зображення;

- тег;

- відмітка про вподобання;

- пошуковий запит.

Схема бази даних наведена у графічному матеріалі.

В таблиці «AspNetUsers» міститься інформація користувачів, необхідна для того, щоб реєструвати та авторизувати користувачів системи. В таблиці знаходяться поля Id, UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnable, AccessFailedCount в яких знаходяться інформація, що необхідна для реєстрації та авторизації користувачів. Для реєстрації користувачів використовується готове програмне забезпечення компанії Microsoft – ASP.Net Core Identity.

Таблиця 2.1 – Опис таблиці «AspNetUsers»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
Id	Унікальний ідентифікатор користувача	nvarchar(450)	X	X	X	
UserName	Ім'я					
Normalized UserName	Нормалізоване ім'я користувача					
Email	Електронна пошта					
Normaized Email	Нормалізована електронна пошта					
EmailConfirmed	Підтвердження електронної пошти					
PasswordHash	Хеш паролю користувача					
SecurityStamp	Відмітка безпеки для авторизації					

Продовження таблиці 2.1

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
ConcurrencyStemp	Відмітка для запобігання подвійного оновлення					
PhoneNumber	Номер телефону					
PhoneNumberConfirmed	Підтвердження номеру телефону					
TwoFactorEnabled	Включення можливості двохфакторної авторизації					
LockoutEnd	Кінець блокування аккаунту					
LockoutEnabled	Включення можливості блокування аккаунту					
AccessFailedCount	Кількість невдалих спроб авторизації					

Сутність «Користувач» міститься в таблиці «user». В таблиці знаходяться поля id, email, username, firstName, lastName, в яких знаходиться основна інформація про користувача.

Таблиця 2.2 – Опис таблиці «user»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
id	Унікальний ідентифікатор користувача	int	X	X	X	
email	Електронна пошта	nvarchar(512)	X	X		
username	Нікнейм	nvarchar(512)				
firstName	Ім'я	nvarchar(512)				
lastName	Прізвище	nvarchar(512)				

Сутність «Пост» міститься в таблиці «post». Постом є завантажене до колекції сервісу зображення користувача, яке є доступним для перегляду. В таблиці знаходяться поля id, file_id, user_id, title, create_date в яких знаходиться основна інформація про завантажене зображення.

Таблиця 2.3 – Опис таблиці «post»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
id	Унікальний ідентифікатор посту	int	X	X	X	
file_id	Зовнішній ключ файлу зображення	int	X			X
user_id	Зовнішній ключ користувача	int	X			X
title	Заголовок посту	nvarchar(512)				
create_date	Дата створення	date				

Сутність «Відмітка про вподобання» міститься в таблиці «like». В таблиці знаходяться поля user_id, post_id, в яких знаходиться інформація про ті пости, які сподобалися користувачу.

Таблиця 2.4 – опис таблиці «like»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
post_id	Складений унікальний ідентифікатор відмітки, зовнішній ключ посту	int	X	X	X	X

Продовження таблиці 2.4

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
user_id	Складений унікальний ідентифікатор відмітки, зовнішній ключ користувача	int	X	X	X	X

Сутність «Зображення» міститься в таблиці «file». В таблиці знаходяться поля id, binaryBaseValue, в яких знаходиться інформація про файл зображення.

Таблиця 2.5 – опис таблиці «file»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
id	Унікальний ідентифікатор файлу	int	X	X	X	
binaryBase Value	Значення base64 завантаженого файлу	nvarchar(max)				

Сутність «Тег» міститься в таблиці «tag». В таблиці знаходяться поля id, name, post_id, в яких знаходиться інформація про теги постів.

Таблиця 2.6 – опис таблиці «tag»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
id	Унікальний ідентифікатор тегу	int	X	X	X	
name	Значення тегу	nvarchar(512)				
post_id	Зовнішній ключ посту	int				X

Сутність «Пошуковий запит» міститься в таблиці «search_request». В таблиці знаходяться поля user_id, tag, date, в яких знаходиться інформація про пошукові запити користувачів.

Таблиця 2.7 – опис таблиці «search_request»

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
user_id	Складений унікальний ідентифікатор запиту, зовнішній ключ користувача	int	X	X	X	X

Продовження таблиці 2.7

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
tag	Складений унікальний ідентифікатор запиту	int	X	X	X	
date	Дата виконання запиту	datetime				

Висновок до розділу

В даному розділі було показано і описано структуру спроектованої реляційної бази даних системи та наведено опис структури основних таблиць. Описані сутності та наведені вхідні дані та вихідні дані системи.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Існує множина користувачів сервісу. Кожен користувач має власну множину пошукових запитів серед зображень із колекції сервісу. Також існує множина постів із зображеннями. Кожен пост має власну множину тегів , що задаються користувачами при створені постів. В ході використання сервісу користувачі виконують пошук зображень за тегами та відмічають ті зображення, які їм сподобалися.

Користувач в більшій мірі буде виконувати пошук та переглядати ті зображення, які йому більш цікаві, ніж інші. Також, користувачі, що шукають та переглядають пости за схожими тегами можуть мати спільнівподобання.

Для формування рекомендацій необхідно оцінювати активність користувачів при користуванні сервісом та, при необхідності, створювати рекомендації для користувачів в залежності від їх вподобань. Дану задачу можна вирішити за допомогою методу колаборативної фільтрації (k-means user-based collaborative filtering).

3.2 Математична постановка задачі

При використанні колаборативної фільтрації для формування рекомендацій необхідно вирішити дві задачі:

1) відшукати n-найближчих за вподобаннями користувачів-сусідів;

2) враховуючи вподобання n-найближчих користувачів-сусідів, сформувати список з q рекомендованих тегів.

Дано:

x – цільовий користувач;

U – множина користувачів-сусідів;

u_i – елемент множини U;

m – розмірність множини P_u;

t – розмірність множини U;

s – розмірність множини P_x;

P – множина шуканих користувачами U та x тегів, де P_u (P_x) множина тегів, шуканих користувачем u (x), а P_{Uij} – j-й тег, який шукав i-й користувач;

L – множина оцінок користувачів, де L_{UiPj} – кількість постів з тегом p_j, які сподобалися користувачу u_i;

n – ширина кола найближчих користувачів – їх кількість;

q – кількість тегів, що необхідно взяти для рекомендацій.

Змінні:

r – вектор вподобань користувача x, r = { L_{XPx1}, L_{XPx2}, …, L_{XPxs} }

v – множина векторів вподобань користувачів-сусідів,

де v_i = {R_{Pui1}, R_{Puj2}, … , R_{Pujs}} – вектор вподобань i-го користувача з множини U;

$R_{P_{u_{ij}}} = \begin{cases} 0, Pu_{ij} \notin Px \\ L_{u_i p_j} + 1, Pu_{ij} \in Px \end{cases}$, оцінка користувача u_i для тега Pu_{ij};

S_{rv_i} – числове значення подібності i-го користувача та цільового користувача;

N – множинна найближчих користувачів-сусідів;

w_i – вага i-го тегу з множини P_n для рекомендації;

$y_i = \begin{cases} 0 \\ 1 \end{cases}$, i-й тег з множини P_n рекомендований або не рекомендований;

P_n – множина нових тегів з тих, що шукали найближчі сусіди

Обмеження:

−обмеження кількості рекомедованих тегів:

$$\sum_{1}^{m} y_i \leq q$$

−подібність користувачів за косинусною мірою:

$$S_{rv_i} = \frac{\sum_k^s r_{ik} v_{jk}}{\sqrt{\sum_k^s r_{ik}^2 \sum_k^s v_{jk}^2}} - \quad (3.1)$$

−вибір n найближчих сусідів:

$$N = \{ k \in U | f_{u_i u_j} = \begin{cases} 0, & S_{rv_i} \geq S_{rv_j} \\ 1, & S_{rv_i} < S_{rv_j} \end{cases}, \sum_{1}^{t} f_{ku_i} \leq n \quad (3.2)$$

− розрахунок ваг тегів для рекомендації:

$$w_i = \frac{\sum_1^n L_{N_i p_{n_j}}}{n} (3.3)$$

Цільова функція:

$$Z = (\sum_{1}^{q} (y_i \times w_i)) \rightarrow max$$

3.3 Обґрунтування методу розв'язання

Для прийняття рішення стосовно обрання алгоритму для реалізації формування рекомендацій для користувачів було розглянуто ряд статей [1-3] У [1-2] автори пропонують використовувати алгоритм колаборативної фільтрації для формування рекомендацій по тегам з використанням класичного методу пошуку k-найближчих сусідів, з певними модифікаціями з урахуванням кількості даних, яку сьогодні необхідно обробляти системам.

У [1] Для розрахунку схожості користувачів пропонується використовувати метод розрахунку величини перетину інтересів користувачів – спільних тегів та оцінених об'єктів.

Автори [3] пропонують використовувати значення косинусу кута між векторами використаних користувачами тегів для розрахунку схожості вподобань користувачів.

Для вирішення проблеми рекомендації контенту для користувачів сервісу було прийнято рішення реалізувати алгоритм колаборативної фільтрації з використанням значення косинусу кута між векторами вподобань користувачів, як значення схожості користувачів.

3.4 Опис методів розв'язання

Покрокова робота алгоритму:

Крок 1. Обрати цільового користувача x для формування рекомендацій

Крок 2. Обрати множину користувачів U, серед яких буде обрано n найближчих, та множини тегів, які ці користувачі шукали

Крок 3. Для користувача x та кожного користувача з U розрахувати вектори вподобань

Крок 4. Для кожного користувача з U розрахувати значення схожості з цільовим користувачем (3.1)

Крок 5. Обрати n користувачів з найбільшим значенням схожості(3.2)

Крок 6. Для кожного тегу розрахувати значення його ваги (3.3)

Крок 7. Обрати q тегів з найбільшим значенням ваги

Крок 8. Кінець алгоритму

На початку необхідно обрати цільового користувача та множину користувачів, серед яких буде обрано n найближчих сусідів.

Для знаходження n-найближчих необхідно для користувачів порахувати їх наближеність за вподобаннями до іншими користувачами.

Спочатку для цільового користувача будується його вектор вподобань. Далі для кожного користувача, з яким він буде порівнюватися будуватиметься вектор відповідності вподобань користувача-сусіда до вподобань цільового.

Далі розраховується косинус кута між векторами вподобань для всіх користувачів, що і використовуватиметься як значення схожості користувачів (3.1).

Після цього результати розрахунків схожості користувачів сортуються за спаданням та формується множина N (3.2) з n користувачів з найбільшим значенням схожості – вони і є n-найближчими сусідами.

Далі необхідно, використовуючи інформацію про вподобання користувачів-сусідів, знайти теги, які можуть бути цікавими цільовому користувачу, тобто пошук тегів для рекомендації буде проводитись серед тих тегів, які не шукав цільовий користувач, але шукали сусіди.

Для цього формується множина тегів P_n, що складається з тих тегів, які шукали користувачі з множини N, але не шукав цільовий користувач. Для кожного тегу, що можна рекомендувати цільовому користувачу, розраховується його вага (3.3).

Далі теги сортуються за спаданням ваги та обираються q найкращих з них – з найбільшою вагою.

Висновок до розділу

В даному розділі було проаналізовано проблему рекомендації контенту користувачам сервісу на основі тегів. Наведено обґрунтування вибору методу колаборативної фільтрації на основі схожості користувачів з пошуком k-найближчих сусідів та використанням розрахунку косинусу кута між векторами вподобань користувачів, як міри схожості користувачів сервісу. Наведено математичну постановку задачі та опис процесу виконання реалізованого алгоритму.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

У таблиці 3.1 перераховано основні засоби, що були використані при розробці.

Таблиця 3.1 – Основні засоби при розробці

Операційна система	Windows 10
Мови програмування	C#, TypeScript
Платформи та інші засоби розробки	ASP. NET Core, Angular, Entity Framework Core, MS SQL Server
Проектування	Enterprise Architect
Контролю версій	Git, Github

Для збереження та роботи з даними системи було використано MS SQL Server. Сервер бази даних розгорнуто на базі операційної системи Windows 10. Windows 10 – операційна система, розроблена Microsoft. Windows 10 має велику популярність у користувачів, розробники часто випускають оновлення, виправляють помилки, вдосконалюють методи захисту персональних даних користувачів. Дана система підтримує роботу великої кількості програмних засобів, частина з яких була використана при розробці системи, як, наприклад, Microsoft Word, Microsoft PowerPoint та MS SQL Server.

Розробка клієнтської частини системи здійснювалася з використанням мови TypeScript та фреймворку Angular 9.

Angular — це платформа і фреймворк для створення односторінкових клієнтських додатків з використанням HTML і TypeScript. Він реалізує основні та додаткові функціональні можливості у вигляді набору бібліотек TypeScript, які ви імпортуєте в свої додатки [6].

Призначений для розробки односторінкових додатків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета — розширення

браузерних застосунків на основі шаблону Модель-вид-контролер (MVC), а також спрощення їх тестування та розробки.

Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка є звичайними змінними JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON-даних.

Розробка серверної частини системи здійснювалося з використанням високорівневої мови програмування C# та платформи ASP .NET Core.

ASP.NET Core – це кросплатформенний, високопродуктивний фреймворк з відкритим вихідним кодом для створення сучасних хмарних додатків, підключених до Інтернету[4]. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Не зважаючи на те, що це новий фреймворк, побудований на новому веб-стеку, ASP.NET Core має високий ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages. В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності.

Entity Framework Core - це полегшена, розширювана, відкрита і кросплатформенна версія популярної технології доступу до даних Entity Framework.

Entity Framework Core може відображати реляційні дані у об'єкти , дозволяючи розробникам .Net працювати з базою даних з використанням об'єктів і усуваючи необхідність в більшій частині коду доступу до даних, який їм зазвичай потрібно написати[5].

Для створення діаграм було використано Enterprise Architect.

Enterprise Architect – це комплексний інструмент аналізу та проектування UML для UML, SysML, BPMN та багатьох інших технологій. Охоплює розробку програмного забезпечення від збору вимог до етапів аналізу, проектування моделей, тестування та технічного обслуговування[7].

Git – система контролю версій. Особливість Git, яка дійсно відрізняє його від майже всіх інших SCM, полягає в його розгалуженій моделі. Git дозволяє вам мати кілька локальних гілок, які можуть бути повністю незалежні один від одного[8].

GitHub – платформа для зручного керування Git-репозиторієм проекту.

4.2 Вимоги до технічного забезпечення

4.2.1 Вимоги до серверу

– процесор с частотою 1,6 ГГц або потужніший;

– ОЗУ об’ємом 1.5 ГБ;

– 10 ГБ доступного простору на жорсткому диску або на SSD;

– жорсткий диск с частотою обертання 5 400 об/хв або еквівалентний йому SSD;

– відеокарта с підтримкою DirectX 11 і розширення екрану 1024x768 або вище.

4.2.2 Вимоги до серверу баз даних

– процесор з частотою 1,6 ГГц або потужніший;

– RAM об’ємом 1 ГБ;

– 1 ГБ доступного простору на жорсткому диску;

– жорсткий диск с частотою обертання 5 400 об/хв або еквівалентний йому SSD;

– операційна система Windows 10.

4.2.3 Вимоги до клієнта

Наявність одного з веб-браузерів:

- Opera 9.6+;

- Chrome 6.0+.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Для розробки серверної частини системи було використано принцип web api. Сервер приймає запити від клієнтської частини системи до якої клієнт може отримати доступ через інтернет-браузер зі свого комп'ютера. Всі процеси обробки даних, пошуку та додавання нових даних виконуються у контролерах сервера.

Діаграма класів (яка розміщена у графічному матеріалі) відображає структуру серверної частини системи. Кожен контролер відповідає за підтримку деякої частини бізнес-процесів системи: у контролерах реалізовано функціонал, що обробляє запити на отримання даних, додавання нових даних або оновлення даних у таблицях бази даних.

4.3.2 Діаграма послідовності

В даному розділі буде наведено опис діаграм послідовності для основних бізнес-процесів розроблюваної системи, а саме: перегляд зображень на основній сторінці веб-сервісу, перегляд рекомендацій та завантаження зображення користувача до колекції сервісу.

Діаграма послідовності процесу завантаження зображень на основній сторінці (яка розміщена у графічному матеріалі) відображає процес отримання інформації про пости із зображеннями при переході користувача на головну сторінку веб-сервісу. При переході на клієнтський застосунок надсилає на back-end сервер запит на отримання постів. На стороні back-end серверу запит обробляє PostController. Він отримує інформацію про користувача та формує і

викликає запит до бази даних для отримання постів для користувача. Після цього інформація про пости передається клієнту та відображається на сторінці.

Діаграма послідовності процесу завантаження рекомендацій (яка розміщена у графічному матеріалі) відображає процес отримання інформації про пости із зображенням, які можна рекомендувати користувачу для перегляду при переході користувача на відповідну сторінку веб-сервісу. При переході на сторінку перегляду рекомендацій, клієнтський застосунок надсилає запит на back-end сервер для отримання рекомендацій. На стороні back-end серверу запит обробляє PostController. Він отримує інформацію про користувача та на її основі за допомогою RecommendationService'у формує множину тегів, які могли б сподобатися користувачу. Після цього контролер формує та викликає запит до бази даних на отримання постів з рекомендованими тегами та відправляє інформацію про пости клієнту. Після отримання даних на стороні клієнта, пости відображаються на сторінці.

Діаграма послідовності процесу завантаження зображення користувача до колекції сервісу (яка розміщена у графічному матеріалі) відображає процес завантаження зображення користувача до колекції сервісу. Клієнт вводить заголовок для посту, теги для його пошуку та обирає зображення для завантаження. Після підтвердження форми, на back-end сервер надсилається запит з інформацією про створений пост. На стороні back-end серверу запит обробляє PostController. Він формує та викликає запити до бази даних на створення посту, тегів та збереження файлу. Після цього клієнту відправляється повідомлення про результат виконання запитів.

4.3.3 Діаграма компонентів

Діаграма компонентів (яка розміщена у графічному матеріалі) відображає основні компоненти розроблюваної системи. За допомогою клієнтської частини користувач взаємодіє з сервером через запити на отримання інформації. Сервер через контролери виконує запити до бази даних на отримання інформації та перетворює її у необхідний формат для

повернення клієнтській частині системи для відображення інформації на сторінці для користувача.

4.3.4 Специфікація функцій

Специфікації функцій представлені в таблицях 4.1 – 4.7

Таблиця 4.1 – Функції класу UserController

Назва	Примітка
`GetUser([FromRoute] int id)`	Отримати інформацію про користувача
`UpdateProfile([FromBody] UpdateProfileModel model)`	Оновити дані профілю користувача

Таблиця 4.2 – Функції класу AccountController

Назва	Примітка
`RegisterAccount([FromBody] RegisterAccountModel model)`	Зареєструвати користувача у системі

Таблиця 4.3 – Функції класу AuthController

Назва	Примітка
`SignIn([FromBody] AuthModel model)`	Зареєструвати користувача у системі

Таблиця 4.4 – Функції класу FileController

Назва	Примітка
`DownloadFileBase64([FromRoute] int fileId)`	Завантажити зображення в форматі base64 з колекції системи

Таблиця 4.5 – Функції класу FileController

Назва	Примітка
`Like([FromBody]LikeModel model)`	Додати позначку, що зображення сподобалося користувачу
`Dislike([FromRoute] int userId, int postId)`	Видалити позначку, що зображення сподобалося користувачу

Таблиця 4.6 – Функції класу SearchController

Назва	Примітка
`AddSearchRequest([FromBody] SearchRequestModel model)`	Зберегти запит користувача на пошук за тегами серед колекції системи

Таблиця 4.7 – Функції класу PostController

Назва	Примітка
`GetUserPosts([FromRoute] int userId, int take, int skip)`	Отримати усі пости, створенні користувачем
`GetPosts([FromRoute]int userId, int take, int skip)`	Отримати пости для відображення у стрічці контенту користувача
`GetRecommendations([FromRoute]int userId, int take, int skip)`	Отримати пости, що можуть бути рекомендовані користувачу для перегляду
`SearchPostsByTags([FromRoute]int userId, string searchKey, int take, int skip)`	Отримати пости, що позначені тегами, введеними користувачем
`DeletePost([FromRoute] int userId, int postId)`	Видалити пост користувача
`CreatePost([FromForm] CreatePostModel model)`	Створити пост

Висновок до розділу

В даному розділі описано програмне забезпечення, фреймворки, платформи, засоби розробки та проектування, що були використані під час розробки системи. Наведено діаграми класів, послідовності та компонентів. Описано функції класів системи.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

В системі фігурує один актор – користувач, зареєстрований у системі. Для користування сервісом, користувачу необхідно зареєструватися у системі, після чого в нього буде змога авторизуватися та користуватися функціоналом сервісу. На рисунку 5.1 представлено форму реєстрації користувача.

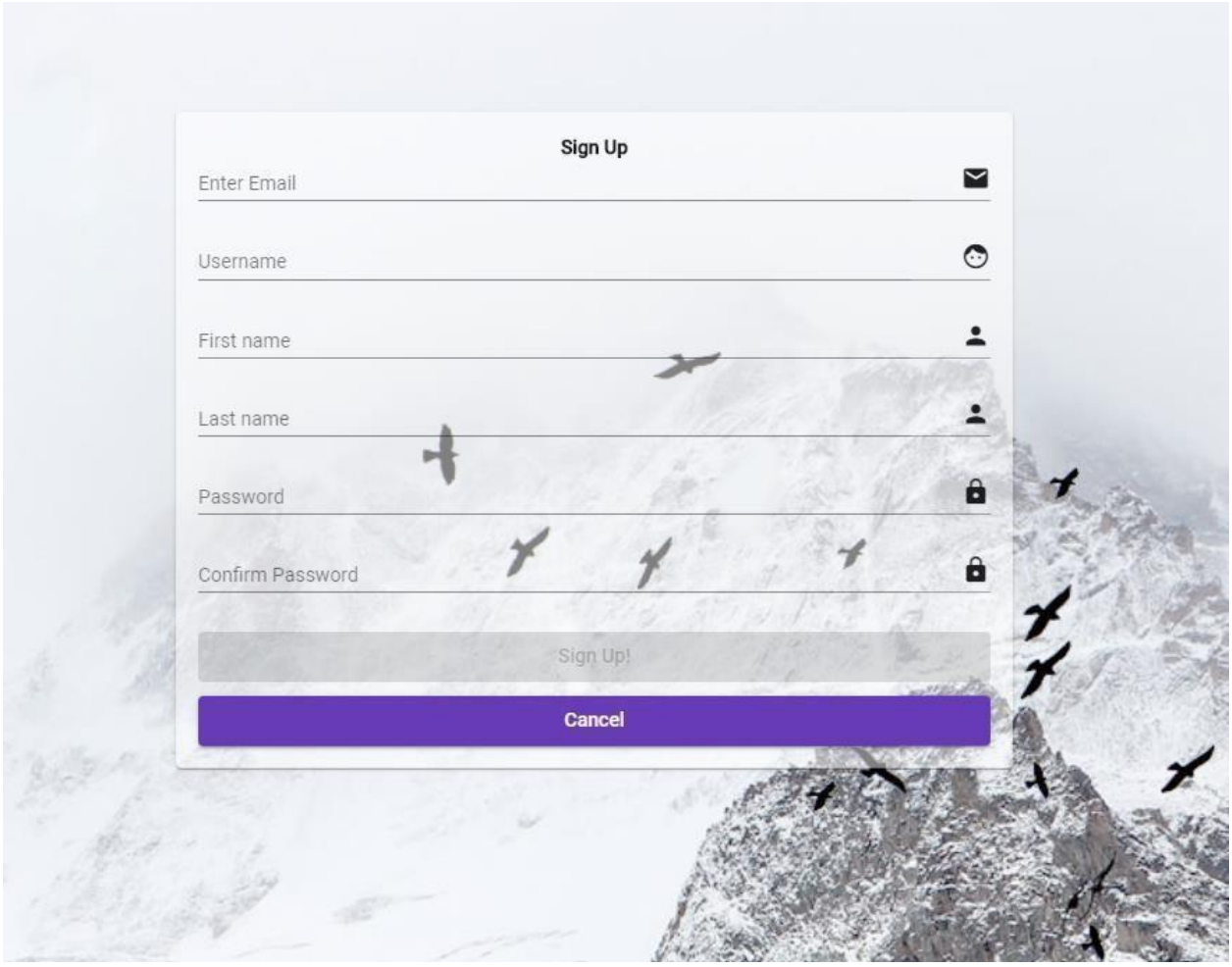


Рисунок 5.1 – Форма реєстрації користувача

Для реєстрації в системі користувачу необхідно ввести унікальну електронну пошту, тобто таку, яку не зареєстровано у системі, нікнейм користувача, ім’я користувача, прізвище користувача, пароль та підтвердження паролю. Після успішної реєстрації користувач зможе авторизуватися у системі. На рисунку 5.2 представлено форму авторизації користувача.

Рисунок 5.2 – Форма авторизації користувача

Для авторизації у системі користувач повинен ввести електронну пошту, яку він використовував при реєстрації та пароль. При невірних введених даних буде виведено відповідне повідомлення.

Після успішної авторизації користувача буде перенаправлено на головну сторінку сервісу. Навігація користувача по сторінках сервісу відбувається за допомогою меню у шапці веб-сервісу. На рисунку 5.3 представлено головну сторінку.

Рисунок 5.3 – Головна сторінка

На головній сторінці для користувача відображаються пости із зображеннями в залежності від активності користувача у сервісі: якщо користувач тільки зареєструвався та ще не виконував пошук зображень серед колекції сервісу, то йому будуть пропонуватися пости, популярні серед інших користувачів. Надалі, якщо користувач виконуватиме пошук зображень за тегами, зображення на головній сторінці фільтруватимуться відповідно до тих тегів, які користувач шукав останніми. Якщо користувач активно шукає зображення серед колекції сервісу (5 та більше унікальних тегів за останній час), то в нього буде можливість переглянути рекомендації – пости із теги, які система вважатиме такими, що можуть сподобатися користувачу, розраховуючи із схожості історії пошуків користувача із історіями пошуку інших користувачів.

Користувач оцінювати ті пости із зображеннями, які їм сподобалися, ставлячи помітку «Сподобалося».

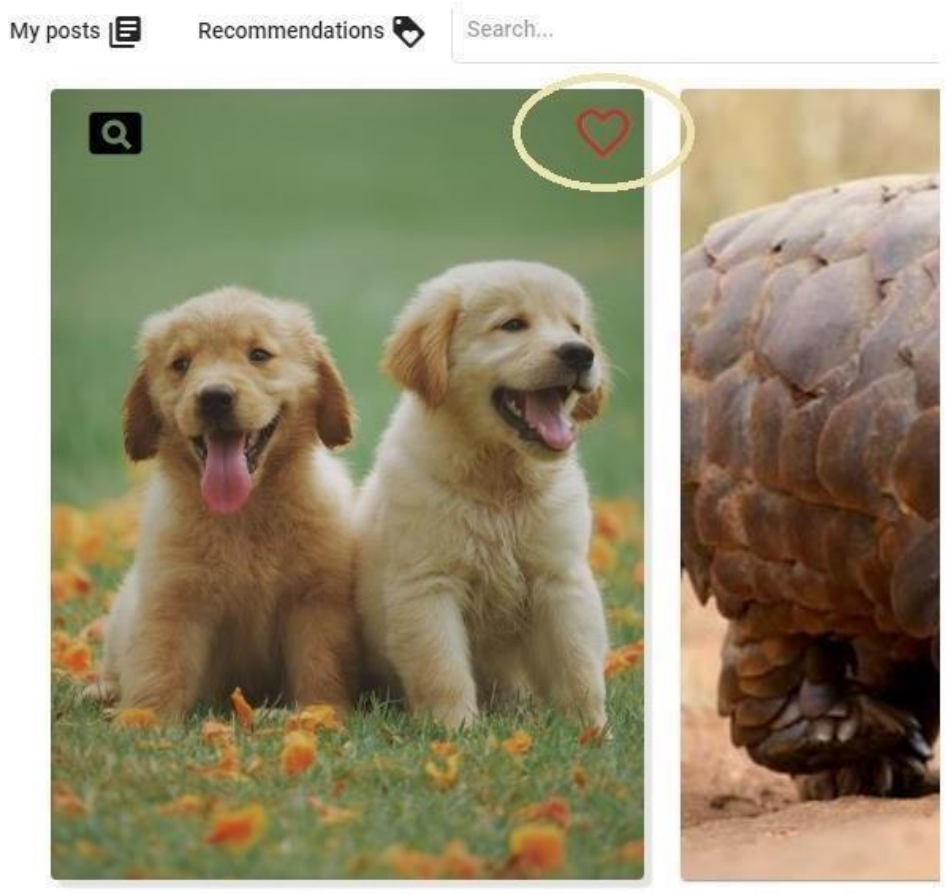


Рисунок 5.4 – Оцінювання постів із зображеннями

Також користувач має змогу переглядати пости у збільшеному форматі, для кращого відображення зображення. У модальному вікні перегляду посту відображається заголовок, обраний автором посту, його нікнейм, повне ім’я та теги, що були обрані для посту. На рисунку 5.5 представлено модальне вікно для перегляду посту.

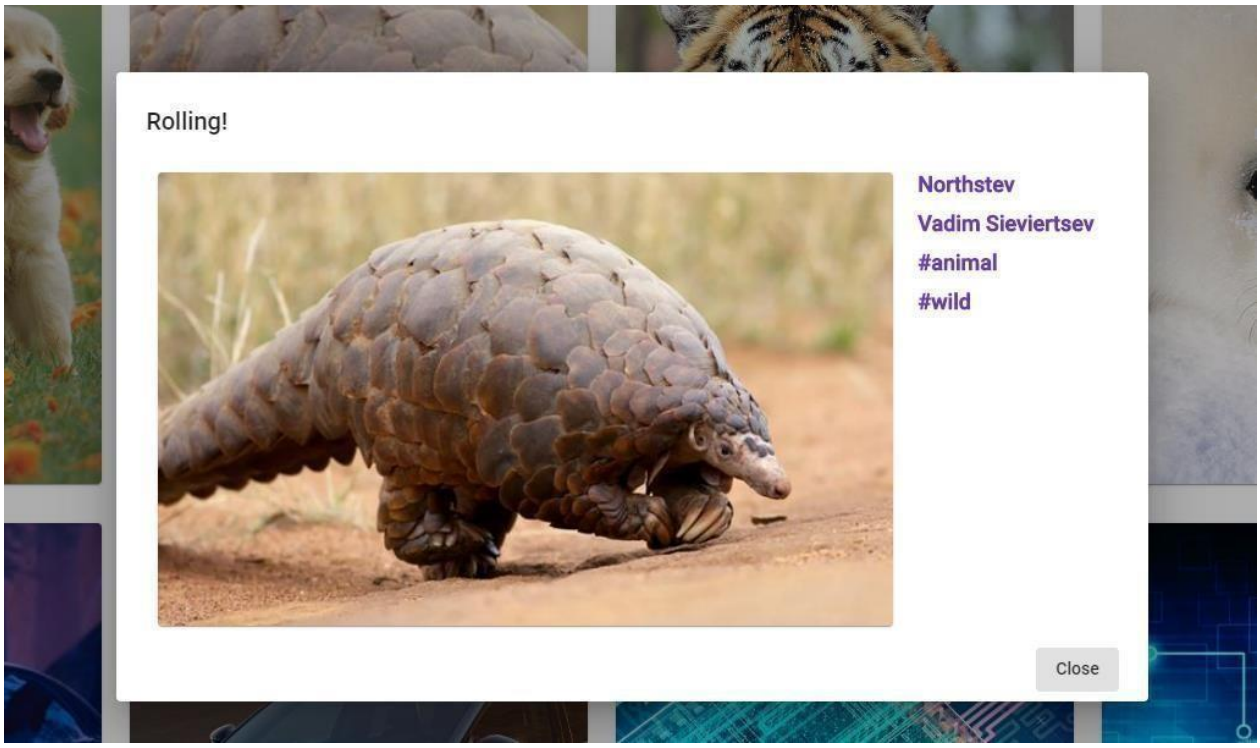


Рисунок 5.5 – Перегляд посту у модальному вікні

Користувач може переглядати пости, які він завантажив до колекції сервісу. На рисунку 5.6 представлено сторінку перегляду постів користувача.

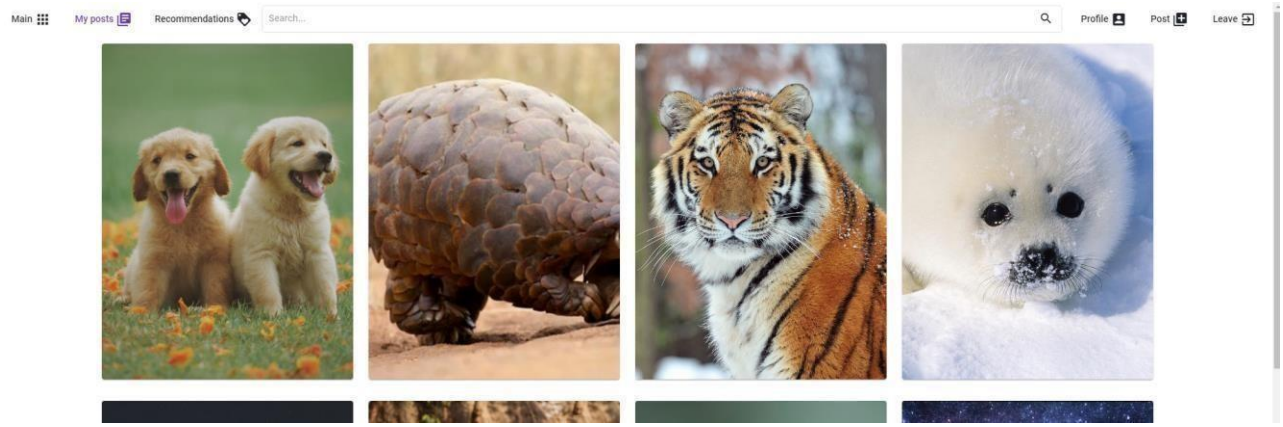


Рисунок 5.6 – Перегляд постів, завантажених користувачем

Користувач має змогу видаляти зображення, завантажені ним до колекції сервісу.

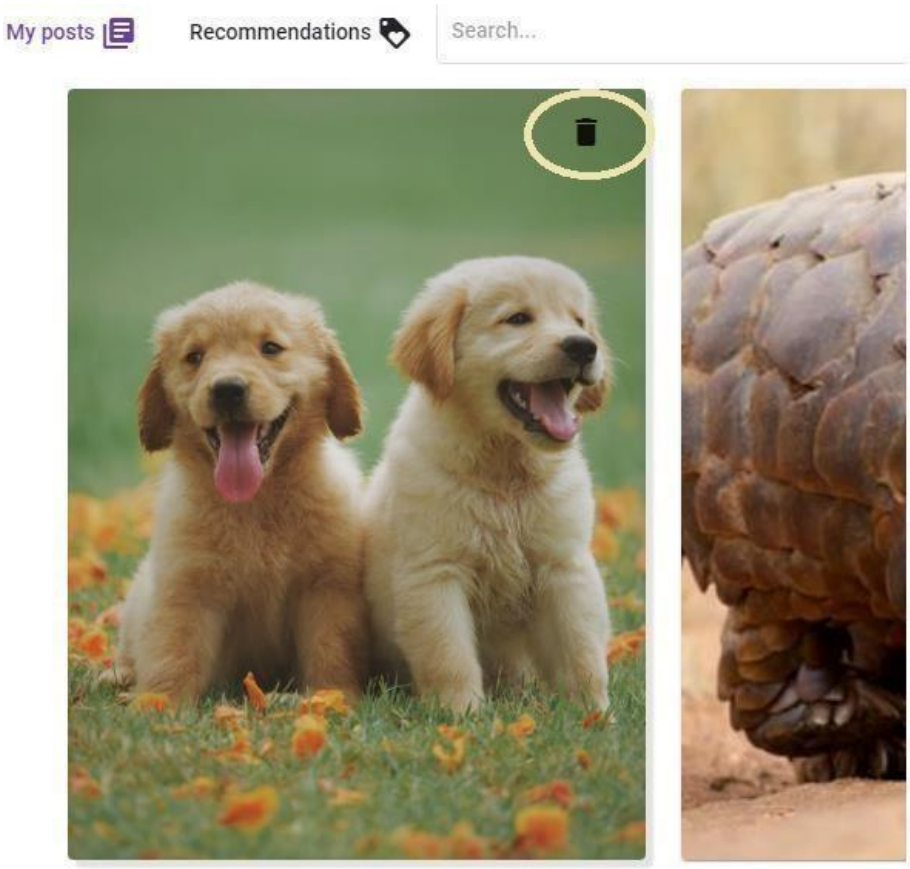


Рисунок 5.7 – Видалення зображення з колекції сервісу

Якщо користувач активно шукає зображень за тегами серед колекції сервісу, у нього є змога переглянути зображення, які система вважатиме схожими з тим, які шукає користувача, основуючись на інтересах користувачів зі схожими історіями пошуку. На рисунку 5.8 представлено сторінку перегляду рекомендацій.

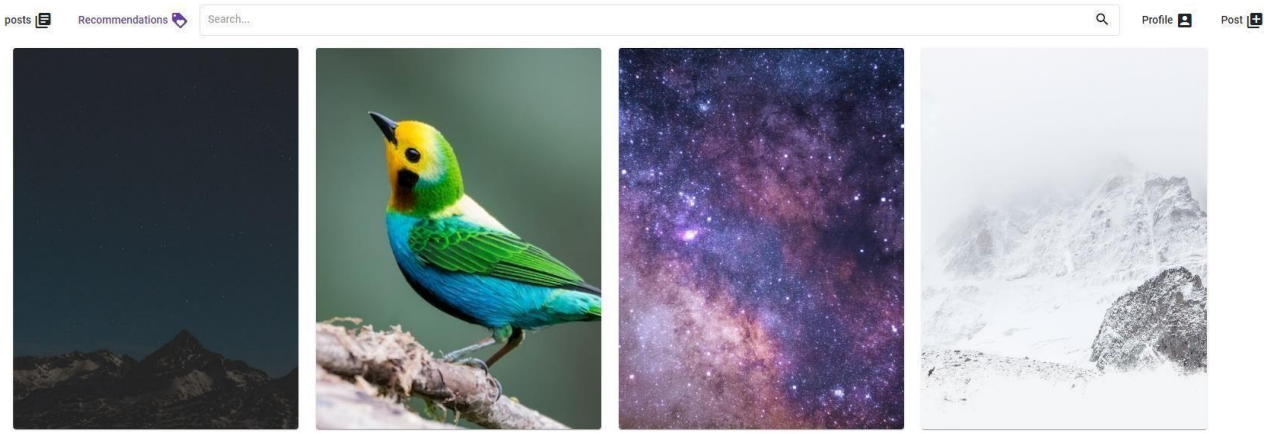


Рисунок 5.8 – Сторінка перегляду рекомендацій

Користувач може виконувати пошук зображень за тегами серед колекції сервісу. Для цього необхідно через пробіл ввести теги за якими необхідно фільтрувати зображення у поле для вводу. Після підтвердження пошуку користувач зможе переглядати відфільтровані за тегами пости із зображеннями. На рисунку 5.9 наведено результат виконання фільтрування постів за тегами.

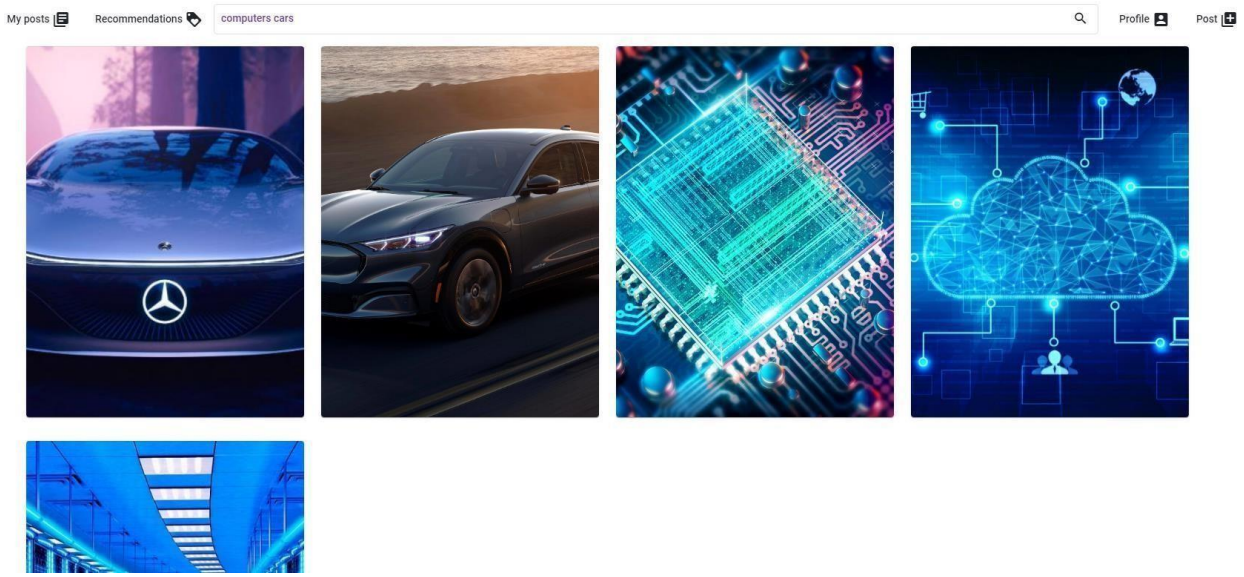


Рисунок 5.9 – Перегляд відфільтрованих постів

Користувач має змогу змінювати дані профілюкористувача. На рисунку 5.10 наведено сторінку з формою для зміни даних профілю.

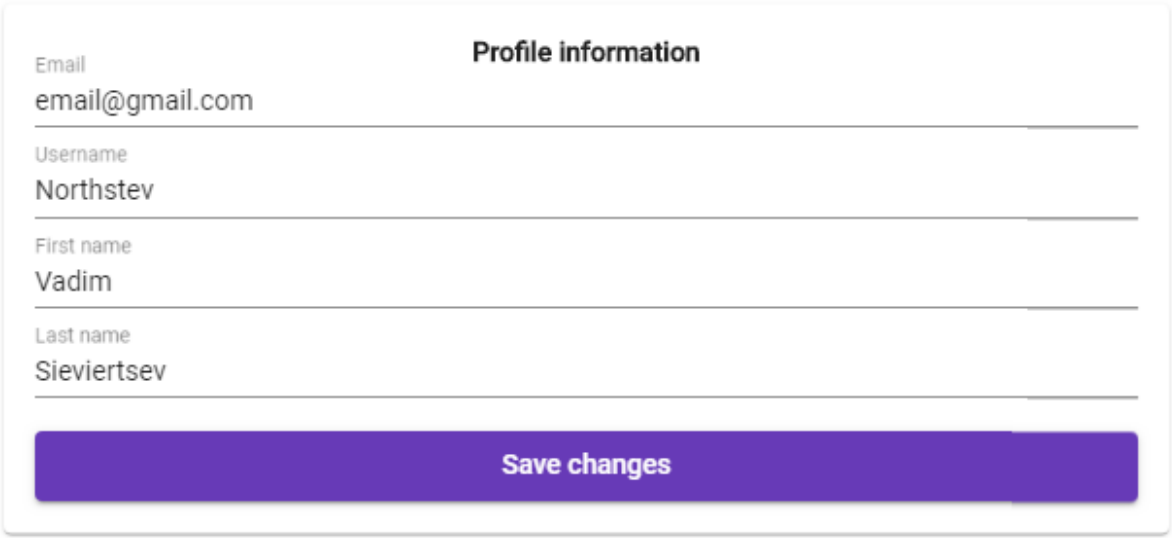


Рисунок 5.10 – Форма редагування даних профілю

Для завантаження зображення до колекції сервісу, користувач повинен ввести заголовок для посту, теги для пошуку та обрати зображення для завантаження. На рисунку 5.11 наведено форму для завантаження зображення та створення посту.

Рисунок 5.11 – Форма створення посту

Для виходу з системи користувач повинен обрати пункт «Leave» з меню у шапці сервісу.

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення функціональним вимогам, представленим у технічному завданні.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій інформаційної системи підтримки сервісу розповсюдження фотознімків вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

- ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Під час тестування були перевірені основні функціональні можливості системи. Випробування описані у таблиця 5.1 – 5.24

Таблиця 5.1 – Перевірка функції реєстрації

Назва	Перевірка функції реєстрації
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю

Продовження таблиці 5.1

Схема проведення тесту	Ввести унікальну електронну пошту у поле «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», ввести пароль у поле «Password». Підтвердити пароль у полі «Confrim password» Натиснути кнопку «Sign Up!»
Очікуваний результат	Повідомлення про успішну реєстрацію та можливість авторизуватися на сторінці авторизації
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.2 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Ввести унікальну електронну пошту у поле «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», ввести пароль у поле «Password». Залишити пустим поле «Confrim password»

Продовження таблиці 5.2

Назва	**Перевірка функції реєстрації**
Очікуваний результат	Повідомлення про необхідність заповнення поля «Confirm Password»
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.3 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Ввести унікальну електронну пошту у поле «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», залишити пустим поле «Password». Залишити пустим поле «Confrim password»
Очікуваний результат	Повідомлення про необхідність заповнення поля «Password»
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.4 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Ввести унікальну електронну пошту у поле «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», ввести пароль у поле «Password». Ввести інший пароль у поле «Confrim password»
Очікуваний результат	Повідомлення про не співпадіння паролів
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.5 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Залишити поле пустим «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», ввести пароль у поле «Password».

Продовження таблиці 5.5

	Підтвердити пароль у полі «Confrim Password»
Очікуваний результат	Повідомлення про необхідність заповнення поля з електронною поштою
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.6 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Ввести зайняту електронну пошту у поле «Email», ввести нікнейм у поле «Nickname» , ввести ім’я у поле «First name», ввести прізвище у поле «Last name», ввести пароль у поле «Password». Ввести інший пароль у поле «Confrim password»
Очікуваний результат	Повідомлення про існування користувача з введеною електронною поштою
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.7 – Перевірка функції реєстрації

Назва	**Перевірка функції реєстрації**
Початковий стан системи	Відкрита сторінка реєстрації
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище, пароль, підтвердження паролю
Схема проведення тесту	Ввести унікальну електронну пошту у поле «Email», залишити пустим поле «Nickname» залишити пустим поле «First name», залишити пустим поле «Last name», ввести пароль у поле «Password». Ввести інший пароль у поле «Confrim password»
Очікуваний результат	Повідомлення про необхідність заповнення для реєстрації полів нікнейму, ім’я та прізвища користувача
Стан системи після проведення випробувань	Відкрита форма реєстрації

Таблиця 5.8 – Перевірка функції авторизації

Назва	**Перевірка функції авторизації**
Початковий стан системи	Відкрита сторінка авторизації
Вхідні дані	Електронна пошта, пароль
Схема проведення тесту	Ввести електронну пошту у поле «Email», ввести пароль у поле «Password». Натиснути кнопку «Sign in»
Очікуваний результат	Перехід на головну сторінку
Стан системи після проведення випробувань	Відкрита головна сторінка

Таблиця 5.9 – Перевірка функції авторизації

Назва	**Перевірка функції авторизації**
Початковий стан системи	Відкрита сторінка авторизації
Вхідні дані	Електронна пошта, пароль
Схема проведення тесту	Залишити пустим поле «Email», ввести пароль у поле «Password». Натиснути кнопку «Sign in»
Очікуваний результат	Повідомлення про необхідність введення електронної пошти
Стан системи після проведення випробувань	Відкрита сторінка авторизації

Таблиця 5.10 – Перевірка функції авторизації

Назва	**Перевірка функції авторизації**
Початковий стан системи	Відкрита сторінка авторизації
Вхідні дані	Електронна пошта, пароль
Схема проведення тесту	Ввести електронну пошту у поле «Email», залишити пустим поле «Password». Натиснути кнопку «Sign in»
Очікуваний результат	Повідомлення про необхідність введення паролю
Стан системи після проведення випробувань	Відкрита сторінка авторизації

Таблиця 5.11 – Перевірка функції авторизації

Назва	**Перевірка функції авторизації**
Початковий стан системи	Відкрита сторінка авторизації
Вхідні дані	Електронна пошта, пароль
Схема проведення тесту	Ввести електронну пошту у поле «Email», ввести неправильний пароль у поле «Password». Натиснути кнопку «Sign in»
Очікуваний результат	Повідомлення про невірно введенні електронну пошту або пароль
Стан системи після проведення випробувань	Відкрита сторінка авторизації

Таблиця 5.12 – Перевірка функції авторизації

Назва	**Перевірка функції авторизації**
Початковий стан системи	Відкрита сторінка авторизації
Вхідні дані	Електронна пошта, пароль
Схема проведення тесту	Ввести неправильну електронну пошту у поле «Email», ввести пароль у поле «Password». Натиснути кнопку «Sign in»
Очікуваний результат	Повідомлення про невірно введенні електронну пошту або пароль
Стан системи після проведення випробувань	Відкрита сторінка авторизації

Таблиця 5.13 – Перевірка функціїрекомендацій

Назва	**Перевірка функції рекомендацій**
Початковий стан системи	Відкрита головна сторінка
Вхідні дані	
Схема проведення тесту	Перейти на сторінку «Recommendations» не виконуючи пошук зображень серед колекції системи
Очікуваний результат	Повідомлення про неможливість формування рекомендацій
Стан системи після проведення випробувань	Відкрита сторінка рекомендацій

Таблиця 5.14 – Перевірка функціїрекомендацій

Назва	**Перевірка функції рекомендацій**
Початковий стан системи	Відкрита головна сторінка
Вхідні дані	
Схема проведення тесту	Перейти на сторінку «Recommendations» виконавши пошук зображень серед колекції системи за п’ятьма або більше тегами
Очікуваний результат	Відображення рекомендованих системою зображень для перегляду
Стан системи після проведення випробувань	Відкрита сторінка рекомендацій

Таблиця 5.15 – Перевірка функції пошуку

Назва	**Перевірка функції пошуку**
Початковий стан системи	Відкрита головна сторінка
Вхідні дані	Теги для пошуку
Схема проведення тесту	Ввести у поле для пошуку теги для пошуку зображень та натиснути значок «Пошук»
Очікуваний результат	Відображення відфільтрованих зображень у стрічці користувача
Стан системи після проведення випробувань	Відкрита головна сторінка

Таблиця 5.16 – Перевірка функції пошуку

Назва	**Перевірка функції пошуку**
Початковий стан системи	Відкрита головна сторінка
Вхідні дані	Теги для пошуку
Схема проведення тесту	Ввести у поле для пошуку теги для яких не існуватиме постів та натиснути значок «Пошук»
Очікуваний результат	Відображення повідомлення про відсутність постів з введеними тегами для пошуку
Стан системи після проведення випробувань	Відкрита головна сторінка

Таблиця 5.17 – Перевірка функції пошуку

Назва	**Перевірка функції пошуку**
Початковий стан системи	Відкрита головна сторінка
Вхідні дані	Теги для пошуку
Схема проведення тесту	Нічого не вводити у поле дляпошуку та натиснути значок «Пошук»
Очікуваний результат	Нічого не відбувається
Стан системи після проведення випробувань	Відкрита головна сторінка

Таблиця 5.18 – Перевірка функції редагування профілю

Назва	**Перевірка функції редагування профілю**
Початковий стан системи	Відкрита сторінка профілю
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище
Схема проведення тесту	Очистити поле «Email», інші поля залишити незмінними
Очікуваний результат	Відображення повідомлення про необхідність заповнення поля «Email» для внесення змін
Стан системи після проведення випробувань	Відкрита сторінка профілю

Таблиця 5.19 – Перевірка функції редагування профілю

Назва	**Перевірка функції редагування профілю**
Початковий стан системи	Відкрита сторінка профілю
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище
Схема проведення тесту	Очистити всі поля
Очікуваний результат	Відображення повідомлення про необхідність заповнення полів «Email», «Username», «First name», «Last name» для внесення змін
Стан системи після проведення випробувань	Відкрита сторінка профілю

Таблиця 5.20 – Перевірка функції редагування профілю

Назва	**Перевірка функції редагування профілю**
Початковий стан системи	Відкрита сторінка профілю
Вхідні дані	Електронна пошта, нікнейм, ім’я, прізвище
Схема проведення тесту	Внести зміни у поле «Email», інші поля залишити незмінними та натиснути кнопку «Confirm»
Очікуваний результат	Відображення повідомлення про результат запиту на внесення змін до даних профілю користувача
Стан системи після проведення випробувань	Відкрита сторінка профілю

Таблиця 5.21 – Перевірка функції завантаження зображення

Назва	**Перевірка функції завантаження зображення**
Початковий стан системи	Відкрита сторінка створення посту
Вхідні дані	Заголовок, теги, зображення
Схема проведення тесту	Ввести заголовок у поле «Title», ввести теги у поле «Tags», обрати зображення для завантаження та натиснути кнопку «Create Post»
Очікуваний результат	Відображення повідомлення про результат запиту на створення посту
Стан системи після проведення випробувань	Відкрита сторінка створення посту

Таблиця 5.22 – Перевірка функції завантаження зображення

Назва	**Перевірка функції завантаження зображення**
Початковий стан системи	Відкрита сторінка створення посту
Вхідні дані	Заголовок, теги, зображення
Схема проведення тесту	Залишити пустим поле «Title», ввести теги у поле «Tags», обрати зображення для завантаження
Очікуваний результат	Відображення повідомлення про необхідність введення заголовку для створення посту
Стан системи після проведення випробувань	Відкрита сторінка створення посту

Таблиця 5.23 – Перевірка функції завантаження зображення

Назва	**Перевірка функції завантаження зображення**
Початковий стан системи	Відкрита сторінка створення посту
Вхідні дані	Заголовок, теги, зображення
Схема проведення тесту	Ввести заголовок у поле «Title», залишити пустим поле «Tags», обрати зображення для заваантаження та натиснути кнопку «Create Post»
Очікуваний результат	Відображення повідомлення про необхідність введення тегів для створення посту
Стан системи після проведення випробувань	Відкрита сторінка створення посту

Таблиця 5.24 – Перевірка функції завантаження зображення

Назва	**Перевірка функції завантаження зображення**
Початковий стан системи	Відкрита сторінка створення посту
Вхідні дані	Заголовок, теги, зображення
Схема проведення тесту	Ввести заголовок у поле «Title», ввести теги у поле «Tags», не обрати зображення для завантаження
Очікуваний результат	Відображення повідомлення про необхідність обрання зображення для створення посту
Стан системи після проведення випробувань	Відкрита сторінка створення посту

Висновок до розділу

В даному розділі було наведено керівництво користувача для користування системою та проведено тестування основних функціональних можливостей системи, а саме: реєстрації, авторизації, перегляду рекомендацій, пошуку зображень, завантаження зображень до колекції сервісу та зміни даних профілю користувача.

ЗАГАЛЬНІ ВИСНОВКИ

В межах дипломного проекту було розроблено веб-сервіс для розповсюдження фотознімків та інформаційну систему для підтримки роботи сервісу та забезпечення всіх необхідних механізмів роботи з інформацією: завантаження, пошуку, фільтрування, збереження, редагування. Розроблено та реалізовано алгоритм для рекомендації зображень для перегляду на основі методу колаборативної фільтрації з пошуком k-найближчих користувачів-сусідів. Було створено керівництво користувача та проведено тестування основних функціональних можливостей системи на відповідність поставленим вимогам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Liang H., Xu Y., Li Y., Nayak R. (2009) Tag Based Collaborative Filtering for Recommender Systems. In: Wen P., Li Y., Polkowski L., Yao Y., Tsumoto S., Wang G. (eds) Rough Sets and Knowledge Technology. RSKT 2009. Lecture Notes in Computer Science, vol 5589. Springer, Berlin, Heidelberg

2. Marinho L.B., Schmidt-Thieme L. (2008) Collaborative Tag Recommendations. In: Preisach C., Burkhardt H., Schmidt-Thieme L., Decker R. (eds) Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, Berlin, Heidelberg

3. F. Yin, X. Liu, W. Ding and R. Zhang, "Tag-Based collaborative filtering recommendation algorithm for TV program," 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, 2016, pp. 47-52, doi: 10.1109/ICCWAMTIP.2016.8079802.

4. ASP .NET Core [Електронний ресурс] / Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1

5. Entity Framework Core [Електронний ресурс] / Режим доступу: https://docs.microsoft.com/en-us/ef/core/

6. Angular [Електронний ресурс] / Режим доступу: https://angular.io/guide/architecture

7. Enterprise Architect [Електронний ресурс] / Режим доступу: https://www.sparxsystems.eu/

8. Git [Електронний ресурс] / Режим доступу: https://git-scm.com/about

Додаток А

Тексти програмного коду

Інформаційна система підтримки сервісу розповсюдження фотознімків

(Найменування програми (документа))

DVD-R

(Вид носія даних)

23 арк, 192 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

Клас ApplicationDbContext:

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace projectx.Contexts
{
    public class ApplicationDbContext : IdentityDbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(GetConnectionString());
        }

        private static string GetConnectionString()
        {
            const string databaseName = "projectx";

            return $"Server=LAPTOP-6R7SBS7T;" +
                   $"Database={databaseName};" +
                   "Trusted_Connection=True;" +
                   "MultipleActiveResultSets = true";
        }
    }
}
```

Клас ProjectxContext:

```
using Microsoft.EntityFrameworkCore;
using projectx.Data;
namespace Projectx.Contexts
{
    public partial class ProjectxContext : DbContext
    {
        public ProjectxContext()
        {
        }

        public projectxContext(DbContextOptions<projectxContext> options)
            : base(options)
        {
        }

        public virtual DbSet<File> File { get; set; }
        public virtual DbSet<Post> Post { get; set; }
        public virtual DbSet<Tag> Tag { get; set; }
        public virtual DbSet<User> User { get; set; }
        public virtual DbSet<Like> Like { get; set; }
        public virtual DbSet<SearchRequest> SearchRequest { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
                optionsBuilder.UseSqlServer("Server=LAPTOP-
6R7SBS7T;Database=projectx;Trusted_Connection=True;MultipleActiveResultSets=true");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<File>(entity =>
            {
                entity.ToTable("file");

                entity.Property(e => e.Id).HasColumnName("id");

                entity.Property(e => e.BinaryBaseValue).HasColumnName("binaryBaseValue");
            });
```

```
modelBuilder.Entity<Post>(entity =>
{
    entity.ToTable("post");

    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.FileId).HasColumnName("file_id");

    entity.Property(e => e.CreateDate)
       .HasColumnName("create_date")
       .HasColumnType("date")
       .HasDefaultValueSql("(getdate())");

    entity.Property(e => e.Title)
        .HasColumnName("title")
        .HasMaxLength(512);

    entity.Property(e => e.UserId).HasColumnName("user_id");

    entity.HasOne(d => d.File)
        .WithMany(p => p.Post)
        .HasForeignKey(d => d.FileId)
        .HasConstraintName("FK_post_file_id_5070F446");

    entity.HasOne(d => d.User)
        .WithMany(p => p.Posts)
        .HasForeignKey(d => d.UserId)
        .HasConstraintName("FK_post_user_id_5165187F");
});

modelBuilder.Entity<Tag>(entity =>
{
    entity.ToTable("tag");

    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.Name)
        .HasColumnName("name")
        .HasMaxLength(512);

    entity.Property(e => e.PostId).HasColumnName("post_id");

    entity.HasOne(d => d.Post)
        .WithMany(p => p.Tags)
        .HasForeignKey(d => d.PostId)
        .OnDelete(DeleteBehavior.Cascade)
        .HasConstraintName("FK_tag_post_id__71D1E811");
});

modelBuilder.Entity<User>(entity =>
{
    entity.ToTable("user");

    entity.HasIndex(e => e.Email)
        .HasName("UQ_user__AB6E616472185579")
        .IsUnique();

    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.Email)
        .HasColumnName("email")
        .HasMaxLength(512);
```

```
    entity.Property(e => e.FirstName)
                        .HasColumnName("firstName")
                        .HasMaxLength(512);

                    entity.Property(e => e.LastName)
                        .HasColumnName("lastName")
                        .HasMaxLength(512);

                    entity.Property(e => e.Username)
                        .HasColumnName("username")
                        .HasMaxLength(512);
                });

                modelBuilder.Entity<Like>(entity =>
                {
                    entity.HasKey(e => new { e.PostId, e.UserId });

                    entity.ToTable("like");

                    entity.Property(e => e.PostId).HasColumnName("post_id");

                    entity.Property(e => e.UserId).HasColumnName("user_id");

                    entity.HasOne(d => d.Post)
                        .WithMany(p => p.Likes)
                        .HasForeignKey(d => d.PostId)
                        .OnDelete(DeleteBehavior.ClientSetNull)
                        .HasConstraintName("FK_like_post_id_74AE54BC");

                    entity.HasOne(d => d.User)
                        .WithMany(p => p.Likes)
                        .HasForeignKey(d => d.UserId)
                        .OnDelete(DeleteBehavior.ClientSetNull)
                        .HasConstraintName("FK_like_user_id_75A278F5");
                });

                modelBuilder.Entity<SearchRequest>(entity =>
                {
                    entity.HasKey(e => new { e.UserId, e.Tag });

                    entity.ToTable("search_request");

                    entity.Property(e => e.UserId).HasColumnName("user_id");

                    entity.Property(e => e.Tag)
                        .HasColumnName("tag")
                        .HasMaxLength(50);

                    entity.Property(e => e.Date)
                        .HasColumnName("date")
                        .HasColumnType("datetime");

                    entity.HasOne(d => d.User)
                        .WithMany(p => p.SearchRequests)
                        .HasForeignKey(d => d.UserId)
                        .OnDelete(DeleteBehavior.ClientSetNull)
                        .HasConstraintName("FK__search_re__user___787EE5A0");
                });
            }
        }
    }
```

Клас AccountsController:

```
using System;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using projectx.Contexts;
using projectx.Data;
using projectx.Models;

namespace projectx.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AccountsController : Controller
    {
        private readonly SignInManager<IdentityUser> _signInManager;
        private readonly UserManager<IdentityUser> _userManager;
        private readonly ApplicationDbContext _identityContext;
        private readonly projectxContext _dbContext;
        public AccountsController(UserManager<IdentityUser> userManager,
            SignInManager<IdentityUser> signInManager,
            IConfiguration configuration,
            ApplicationDbContext context,
            projectxContext dbContext)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _identityContext = context;
            _dbContext = dbContext;
        }

        [HttpPost]
        [Route("register")]
        public async Task<IActionResult> RegisterAccount([FromBody] RegisterAccountModel
model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            using(var transaction = _dbContext.Database.BeginTransaction())
            {
                try
                {
                    var userIdentity = new IdentityUser
                    {
                        Email = model.Email,
                        EmailConfirmed = true,
                        UserName = model.Email
                    };

                    var user = new User
                    {
                        Email = model.Email,
                        Username = model.Username,
                        FirstName = model.FirstName,
                        LastName = model.LastName
                    };

                    var result = await _userManager.CreateAsync(userIdentity,
model.Password);
```

```
                    if (!result.Succeeded) return BadRequest(new { errors = result.Errors
});

                    await _dbContext.User.AddAsync(user);

                    await _dbContext.SaveChangesAsync();

                    transaction.Commit();
                }
                catch (Exception e)
                {
                    return BadRequest(e);
                }
            }

            return Ok();
        }
    }
}
```

Клас AuthController:

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using projectx.Contexts;
using projectx.Helpers;
using projectx.Models;
namespace projectx.Controllers
{

    [ApiController]
    [Route("api/[controller]")]
    public class AuthController : Controller
    {
        private readonly SignInManager<IdentityUser> _signInManager;
        private readonly UserManager<IdentityUser> _userManager;
        private readonly IConfiguration _configuration;
        private readonly ApplicationDbContext _identityContext;
        private readonly projectxContext _dbContext;
        public AuthController(UserManager<IdentityUser> userManager,
            SignInManager<IdentityUser> signInManager,
            IConfiguration configuration,
            ApplicationDbContext identityContext,
            projectxContext dbContext)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _configuration = configuration;
            _identityContext = identityContext;
            _dbContext = dbContext;
        }

        [HttpPost]
        [Route("signin")]
        public async Task<IActionResult> SignIn([FromBody] AuthModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var identityUser = await _userManager.FindByEmailAsync(model.Email);
```

```
            if (identityUser == null) return BadRequest("Failed on sign in. Password or
email incorect.");

            // check the credentials
            var result = await _signInManager.CheckPasswordSignInAsync(identityUser,
model.Password, false);

            if (!result.Succeeded)
            {
                return BadRequest("Failed on sign in. Password or email incorect.");
            }

            var user = _dbContext.User.FirstOrDefault(u => u.Email == model.Email);

            if(user == null)
            {
                return BadRequest("Cannot find user with such an email");
            }

            var token = TokenFactory.GenerateJwtToken(user, "cutomer",
_configuration["JwtIssuer"], _configuration["JwtAudience"], _configuration["JwtKey"]);
            return Ok(new { token });
        }
    }
}
```

Клас FileController:

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using projectx.Contexts;

namespace projectx.Controllers
{
    [ApiController]
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    public class FileController : Controller
    {
        private readonly projectxContext _dbContext;
        public FileController(projectxContext dbContext)
        {
            _dbContext = dbContext;
        }

        [HttpGet]
        [Route("{fileId}")]
        public async Task<IActionResult> DownloadFileBase64([FromRoute] int fileId)
        {
            var file = await _dbContext.File.FirstOrDefaultAsync(f => f.Id == fileId);

            if (file == null) return BadRequest("No file found");
            var base64 = file.BinaryBaseValue;
            return Ok(new { base64 });
        }
    }
}
```

Клас LikeController:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using projectx.Contexts;
using projectx.Data;
using projectx.Models;

namespace projectx.Controllers
{
    [ApiController]
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    public class LikeController : Controller
    {
        private readonly projectxContext _dbContext;
        public LikeController(projectxContext dbContext)
        {
            _dbContext = dbContext;
        }

        [HttpPost]
        public async Task<IActionResult> Like([FromBody]LikeModel model)
        {
            var like = new Like
            {
                UserId = model.UserId,
                PostId = model.PostId
            };

            await _dbContext.Like.AddAsync(like);
            await _dbContext.SaveChangesAsync();

            return Ok();
        }

        [HttpDelete]
        [Route("{userId}/{postId}")]
        public async Task<IActionResult> Dislike([FromRoute] int userId, int postId)
        {
            var like = await _dbContext.Like.FirstOrDefaultAsync(l => l.UserId == userId
&& l.PostId == postId);

            if (like == null) return BadRequest("No like for such post found");

            _dbContext.Like.Remove(like);
            await _dbContext.SaveChangesAsync();

            return Ok();
        }
    }
}
```

Клас PostController:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using projectx.Contexts;
using projectx.Data;
using projectx.Models;
using  projectx.Services;

namespace projectx.Controllers
{
    [ApiController]
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    public class PostController : Controller
    {
        private readonly RecommendationService _recommendationService;
        private readonly projectxContext _dbContext;
        public PostController(projectxContext dbContext,
                              RecommendationService recommendationService)
        {
            _dbContext = dbContext;
            _recommendationService = recommendationService;
        }

        [HttpGet]
        [Route("all/user/{userId}/{take}/{skip}")]
        public async Task<IActionResult> GetUserPosts([FromRoute] int userId, int take,
int skip)
        {
            var posts = await _dbContext.Post.Where(p => p.UserId ==
userId).OrderByDescending(p => p.CreateDate).Skip(skip).Take(take).ToArrayAsync();

            return Ok(new { posts });
        }

        [HttpGet]
        [Route("all/{userId}/{take}/{skip}")]
        public async Task<IActionResult> GetPosts([FromRoute]int userId, int take, int
skip)
        {
            PostExtendedModel[] posts;
            var user = await _dbContext.User.Include(u =>
u.SearchRequests).FirstOrDefaultAsync(u => u.Id == userId);
            if (user.SearchRequests.Count == 0)
            {
                posts = await _dbContext.Post.OrderByDescending(p =>
p.Likes.Count()).ThenByDescending(p => p.CreateDate).Skip(skip).Take(take).Select(p =>
new PostExtendedModel
                {
                    Id = p.Id,
                    FileId = p.FileId,
                    Title = p.Title,
                    UserId = p.UserId,
                    Liked = p.Likes.Any(l => l.UserId == userId),
                    CreateDate = p.CreateDate,
                    User = new User
                    {
                        Username = p.User.Username,
```

```
FirstName = p.User.FirstName,
                        LastName = p.User.LastName
                    },
                    Tags = p.Tags.Select(t => t.Name).ToList()
                }).ToArrayAsync();

                return Ok(new { posts });
            }

            posts = await _dbContext.Post.OrderByDescending(p => p.Tags.Any(t =>
user.SearchRequests.Any(s => s.Tag.ToLower() == t.Name.ToLower())))
                                         .ThenByDescending(p =>
p.CreateDate).Skip(skip).Take(take).Select(p => new PostExtendedModel
            {
                Id = p.Id,
                FileId = p.FileId,
                Title = p.Title,
                UserId = p.UserId,
                Liked = p.Likes.Any(l => l.UserId == userId),
                CreateDate = p.CreateDate,
                User = new User
                {
                    Username = p.User.Username,
                    FirstName = p.User.FirstName,
                    LastName = p.User.LastName
                },
                Tags = p.Tags.Select(t => t.Name).ToList()
            }).ToArrayAsync();

            return Ok(new { posts });

        }

        [HttpGet]
        [Route("recommendations/{userId}/{take}/{skip}")]
        public async Task<IActionResult> GetRecommendations([FromRoute]int userId, int
take, int skip)
        {
            PostExtendedModel[] posts;
            var user = await _dbContext.User.Include(u =>
u.SearchRequests).FirstOrDefaultAsync(u => u.Id == userId);

            if (user.SearchRequests.Count() < 5)
            {
                posts = new PostExtendedModel[] { };
                return Ok(new { posts });
            }

            var recommendedTags = await
_recommendationService.GetRecommendedTags(userId);

            if (recommendedTags.Count() == 0)
            {
                posts = new PostExtendedModel[] { };
                return Ok(new { posts });
            }

            posts = await _dbContext.Post.Where(p => p.Tags.Any(t =>
recommendedTags.Any(r => r.Name.ToLower() == t.Name.ToLower())))
                                         .OrderByDescending(p =>
p.CreateDate).Skip(skip).Take(take).Select(p => new PostExtendedModel
            {
                Id = p.Id,
                FileId = p.FileId,
                Title = p.Title,
```

```
                UserId = p.UserId,
                Liked = p.Likes.Any(l => l.UserId == userId),
                CreateDate = p.CreateDate,
                User = new User
                {
                    Username = p.User.Username,
                    FirstName = p.User.FirstName,
                    LastName = p.User.LastName
                },
                Tags = p.Tags.Select(t => t.Name).ToList()
            }).ToArrayAsync();

            return Ok(new { posts });
        }

        [HttpGet]
        [Route("search/{userId}/{searchKey}/{take}/{skip}")]
        public async Task<IActionResult> SearchPostsByTags([FromRoute]int userId, string
searchKey, int take, int skip)
        {
            var tags = searchKey.Split(' ').ToArray();
            var posts = await _dbContext.Post.Where(p => p.Tags.Any(t =>
tags.Contains(t.Name))).OrderByDescending(p =>
p.CreateDate).Skip(skip).Take(take).Select(p => new PostExtendedModel
            {
                Id = p.Id,
                FileId = p.FileId,
                Title = p.Title,
                UserId = p.UserId,
                Liked = p.Likes.Any(l => l.UserId == userId),
                CreateDate = p.CreateDate,
                User = new User
                {
                    Username = p.User.Username,
                    FirstName = p.User.FirstName,
                    LastName = p.User.LastName
                },
                Tags = p.Tags.Select(t => t.Name).ToList()
            }).ToArrayAsync();
            return Ok(new { posts });
        }

        [HttpDelete]
        [Route("delete/{userId}/{postId}")]
        public async Task<IActionResult> DeletePost([FromRoute] int userId, int postId)
        {
            var post = await _dbContext.Post.FirstOrDefaultAsync(p => p.UserId == userId
&& p.Id == postId);

            if (post == null)
            {
                return BadRequest("No post found");
            }

            var posts = _dbContext.Post.Remove(post);

            await _dbContext.SaveChangesAsync();

            return Ok();
        }

        [HttpPost]
        public async Task<IActionResult> CreatePost([FromForm] CreatePostModel model)
        {
            var imageBase64 = "";
```

```
            using (var ms = new MemoryStream())
            {
                model.Image.CopyTo(ms);
                var fileBytes = ms.ToArray();
                imageBase64 = Convert.ToBase64String(fileBytes);
            }

            var tagsValues = model.Tags.Split(' ');
            var file = new Data.File
            {
                BinaryBaseValue = imageBase64
            };

            await _dbContext.File.AddAsync(file);

            var post = new Post
            {
                UserId = model.Id,
                Title = model.Title
            };

            await _dbContext.Post.AddAsync(post);

            post.FileId = file.Id;

            var tags = new List<Tag>();

            foreach (var value in tagsValues)
            {
                tags.Add(new Tag
                {
                    Name = value,
                    PostId = post.Id
                });
            }

            await _dbContext.Tag.AddRangeAsync(tags);
            await _dbContext.SaveChangesAsync();
            return Ok();
        }
    }
}
```

Клас SearchController:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using projectx.Contexts;
using projectx.Data;
using projectx.Models;

namespace projectx.Controllers
{
    [ApiController]
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    public class SearchController : Controller
    {
        private readonly projectxContext _dbContext;
        public SearchController(projectxContext dbContext)
```

```
{
_dbContext = dbContext;
        }

        [HttpPost]
        public async Task<IActionResult> AddSearchRequest([FromBody] SearchRequestModel
model)
        {
            var searchResults = await _dbContext.SearchRequest.Where(r => r.UserId ==
model.UserId).ToArrayAsync();

            var tags = model.TagString.Split(' ');
            var searchRequests = new List<SearchRequest>();

            foreach(var tag in tags)
            {
                if(!searchResults.Any(s=>s.Tag == tag))
                {
                    searchRequests.Add(new SearchRequest
                    {
                        UserId = model.UserId,
                        Tag = tag,
                        Date = DateTime.UtcNow
                    });
                }
            }

            await _dbContext.SearchRequest.AddRangeAsync(searchRequests);
            await _dbContext.SaveChangesAsync();

            return Ok();
        }
    }
}
```

Клас UserController:

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using  Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using projectx.Contexts;
using projectx.Helpers;
using projectx.Models;

namespace projectx.Controllers
{

    [Route("api/[controller]")]
    [Authorize(AuthenticationSchemes = "Bearer")]
    [ApiController]
    public class UserController : Controller
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly IConfiguration _configuration;
        private readonly ApplicationDbContext _identityContext;
        private readonly projectxContext _dbContext;
        private readonly MapHelper _mapper;

        public UserController(UserManager<IdentityUser> userManager,
            SignInManager<IdentityUser> signInManager,
            IConfiguration configuration,
```

```
ApplicationDbContext context,
            projectxContext dbContext)

{
            _userManager = userManager;
            _configuration = configuration;
            _identityContext = context;
            _dbContext = dbContext;
            _mapper = new MapHelper();
        }

        [HttpGet]
        [Route("{id}")]
        public async Task<IActionResult> GetUser([FromRoute] int id)
        {
            var user = await _dbContext.User.FirstOrDefaultAsync(u => u.Id == id);

            if (user == null)
            {
                return BadRequest("Invalid user");
            }

            return Ok(user);
        }

        [HttpPost]
        [Route("update")]
        public async Task<IActionResult> UpdateProfile([FromBody] UpdateProfileModel
model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var user = await _dbContext.User.FirstOrDefaultAsync(u => u.Id == model.Id);

            if(user == null)
            {
                return BadRequest("Invalid user");
            }

            var identityUser = await _userManager.FindByEmailAsync(user.Email);

            if(identityUser == null)
            {
                return BadRequest("Invalid user");
            }

            _mapper.MapChanges(model, user);
            await _dbContext.SaveChangesAsync();
            identityUser.Email = model.Email;
            identityUser.UserName = model.Username;
            await  _identityContext.SaveChangesAsync();

            var token = TokenFactory.GenerateJwtToken(user, "cutomer",
_configuration["JwtIssuer"], _configuration["JwtAudience"], _configuration["JwtKey"]);
            return Ok(new { token });
        }
    }
}
```

Клас File:

```
using System;
using System.Collections.Generic;

namespace projectx.Data
{
    public partial class File
    {
        public File()
        {
            Post = new HashSet<Post>();
        }

        public int Id { get; set; }
        public string BinaryBaseValue { get; set; }

        public ICollection<Post> Post { get; set; }
    }
}
```

Клас Like:

```
using System;
using System.Collections.Generic;

namespace projectx.Data
{
    public partial class Like
    {
        public int PostId { get; set; }
        public int UserId { get; set; }

        public Post Post { get; set; }
        public User User { get; set; }
    }
}
```

Клас Post:

```
using System;
using System.Collections.Generic;

namespace projectx.Data
{
    public partial class Post
    {
        public Post()
        {
            Tags = new HashSet<Tag>();
        }

        public int Id { get; set; }
        public int FileId { get; set; }
        public int UserId { get; set; }
        public string Title { get; set; }
        public DateTime CreateDate { get; set; }

        public File File { get; set; }
        public User User { get; set; }
        public ICollection<Like> Likes { get; set; }
        public ICollection<Tag> Tags { get; set; }
    }
}
```

Клас SearchRequest:

```
using System;
using System.Collections.Generic;

namespace projectx.Data
{
    public partial class SearchRequest
    {
        public int UserId { get; set; }
        public string Tag { get; set; }
        public DateTime Date { get; set; }

        public User User { get; set; }
    }
}
```

Клас Tag:

```
namespace projectx.Data
{
    public partial class Tag
    {
        public int Id { get; set; }
        public int PostId { get; set; }
        public string Name { get; set; }

        public Post Post { get; set; }
    }
}
```

Клас User:

```
using System;
using System.Collections.Generic;

namespace projectx.Data
{
    public partial class User
    {
        public User()
        {
            Posts = new HashSet<Post>();
        }

        public int Id { get; set; }
        public string Email { get; set; }
        public string Username { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

        public ICollection<Like> Likes { get; set; }
        public ICollection<Post> Posts { get; set; }
        public ICollection<SearchRequest> SearchRequests { get; set; }
    }
}
```

Клас Mapper:

```
using AutoMapper;

namespace projectx.Helpers
{
    public class MapHelper
    {
        public TypeToMapTo Map<TypeToMapFrom, TypeToMapTo>(TypeToMapFrom model)
        {
            var mapper = new MapperConfiguration(cfg => { cfg.CreateMap<TypeToMapFrom,
TypeToMapTo>(); }).CreateMapper();

            return mapper.Map<TypeToMapFrom, TypeToMapTo>(model);
        }

        public TypeToMapTo[] MapCollection<TypeToMapFrom, TypeToMapTo>(TypeToMapFrom[]
model)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<TypeToMapFrom,
TypeToMapTo>()).CreateMapper();

            return mapper.Map<TypeToMapFrom[], TypeToMapTo[]>(model);
        }

        public void MapChanges<TypeToMapFrom, TypeToMapTo>(TypeToMapFrom source,
TypeToMapTo destination)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<TypeToMapFrom,
TypeToMapTo>()).CreateMapper();

            mapper.Map(source, destination);
        }

        public void MapRangeChanges<TypeToMapFrom, TypeToMapTo>(TypeToMapFrom[] source,
TypeToMapTo[] destination)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<TypeToMapFrom,
TypeToMapTo>()).CreateMapper();

            mapper.Map(source, destination);
        }

        public void MapNestedChanges<OuterTypeToMapFrom, OuterTypeToMapTo,
                                     FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo,
                                     SecondInnerTypeToMapFrom,
SecondInnerTypeToMapTo>(OuterTypeToMapFrom source, OuterTypeToMapTo destination)
        {
            var mapper = new MapperConfiguration(cfg =>
            {
                cfg.CreateMap<OuterTypeToMapFrom, OuterTypeToMapTo>();
                cfg.CreateMap<FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo>();
                cfg.CreateMap<SecondInnerTypeToMapFrom, SecondInnerTypeToMapTo>();
            }
           ).CreateMapper();

            mapper.Map(source, destination);
        }

        public OuterTypeToMapTo NestedMap<OuterTypeToMapFrom, OuterTypeToMapTo,
InnerTypeToMapFrom, InnerTypeToMapTo>(OuterTypeToMapFrom source)
        {
            var mapper = new MapperConfiguration(cfg =>
            {
                cfg.CreateMap<OuterTypeToMapFrom, OuterTypeToMapTo>();
```

```
cfg.CreateMap<InnerTypeToMapFrom, InnerTypeToMapTo>();
            }
            ).CreateMapper();

            return mapper.Map<OuterTypeToMapFrom, OuterTypeToMapTo>(source);
        }

        public OuterTypeToMapTo NestedMap<OuterTypeToMapFrom, OuterTypeToMapTo,
                                          FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo,
                                          SecondInnerTypeToMapFrom,
SecondInnerTypeToMapTo>(OuterTypeToMapFrom source)
        {
            var mapper = new MapperConfiguration(cfg =>
            {
                cfg.CreateMap<OuterTypeToMapFrom, OuterTypeToMapTo>();
                cfg.CreateMap<FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo>();
                cfg.CreateMap<SecondInnerTypeToMapFrom, SecondInnerTypeToMapTo>();
            }
            ).CreateMapper();

            return mapper.Map<OuterTypeToMapFrom, OuterTypeToMapTo>(source);
        }

        public OuterTypeToMapTo[] NestedMapCollection<OuterTypeToMapFrom,
OuterTypeToMapTo,
                                        FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo,
                                        SecondInnerTypeToMapFrom,
SecondInnerTypeToMapTo>(OuterTypeToMapFrom[] source)
        {
            var mapper = new MapperConfiguration(cfg =>
            {
                cfg.CreateMap<OuterTypeToMapFrom, OuterTypeToMapTo>();
                cfg.CreateMap<FirstInnerTypeToMapFrom, FirstInnerTypeToMapTo>();
                cfg.CreateMap<SecondInnerTypeToMapFrom, SecondInnerTypeToMapTo>();
            }
            ).CreateMapper();

            return mapper.Map<OuterTypeToMapFrom[], OuterTypeToMapTo[]>(source);
        }
    }
}
```

Клас TokenFactory:

```
using Microsoft.IdentityModel.Tokens;
using projectx.Data;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace projectx.Helpers
{
    public static class TokenFactory
    {
        public static string GenerateJwtToken(User user,string role, string issuer,
string audience, string key)
        {
            var claims = new List<Claim>
            {
                new Claim("email", user.Email),
                new Claim("id", user.Id.ToString()),
                new Claim("role", role)
            };
```

```
 var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));
            var creds = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                issuer,
                audience,
                claims,
                expires: DateTime.Now.AddMinutes(60),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

Клас AccountControllerModels:

```
using System.ComponentModel.DataAnnotations;
namespace projectx.Models
{
    public class RegisterAccountModel
    {
        [Required]
        public string Email { get; set; }
        [Required]
        public string Username { get; set; }
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

Клас AuthControllerModels:

```
using System.ComponentModel.DataAnnotations;
namespace projectx.Models
{
    public class AuthModel
    {
        [Required]
        public string Email { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

Клас LikeControllerModels:

```
using System.ComponentModel.DataAnnotations;
namespace projectx.Models
{
    public class LikeModel
    {
        [Required]
        public int UserId { get; set; }

        [Required]
        public int PostId { get; set; }
    }
}
```

Клас PostControllerModels:

```
using Microsoft.AspNetCore.Http;
using projectx.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace projectx.Models
{
    public class CreatePostModel
    {
        [Required]
        public int Id { get; set; }
        [Required]
        public IFormFile Image { get; set; }
        public string Title { get; set; }
        public string Tags { get; set; }
    }

    public class PostExtendedModel
    {
        public int Id { get; set; }
        public int FileId { get; set; }
        public int UserId { get; set; }
        public string Title { get; set; }
        public DateTime CreateDate { get; set; }
        public bool Liked { get; set; }
        public User User { get; set; }
        public List<string> Tags { get; set; }
    }
}
```

Клас RecommendationServiceModels:

```
namespace projectx.Models
{
    public class NeighborUser
    {
        public int UserId { get; set; }
        public double CosineSimilarity { get; set; }
    }

    public class TagScore
    {
        public string Name { get; set; }
        public double Score { get; set; }
    }
}
```

Клас SearchControllerModels:

```
namespace projectx.Models
{
    public class SearchRequestModel
    {
        public int UserId { get; set; }
        public string TagString { get; set; }
    }
}
```

Клас UserControllerModels:

```
using System.ComponentModel.DataAnnotations;

namespace projectx.Models
{
    public class UpdateProfileModel
    {
        [Required]
        public int Id { get; set; }
        [Required]
        public string Email { get; set; }
        [Required]
        public string Username { get; set; }
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
    }
}
```

Клас RecommendationService:

```
using Microsoft.EntityFrameworkCore;
using projectx.Contexts;
using projectx.Data;
using projectx.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace projectx.Services
{
    public class RecommendationService
    {
        private readonly projectxContext _dbContext;
        public RecommendationService(projectxContext dbContext)
        {
            _dbContext = dbContext;
        }

        public async Task<Tag[]> GetRecommendedTags(int userId)
        {
            var targetUser = await
_dbContext.User.Include(u=>u.SearchRequests).Include(u=>u.Likes).ThenInclude(l=>l.Post).F
irstOrDefaultAsync(u => u.Id == userId);
            var users = await _dbContext.User.Take(100).Select(u => new User
            {
                Id = u.Id,
                SearchRequests = u.SearchRequests.OrderByDescending(s =>
s.Date).Take(10).ToList(),
                Likes = u.Likes.Select(l => new Like
                {
                    PostId = l.PostId,
                    UserId = l.UserId,
                    Post = new Post
                    {
                        Tags = l.Post.Tags
                    }
                }).ToList()
            }).ToArrayAsync();

            var neighbors = await GetNearestNeighbors(targetUser);
```

```
            var tags = neighbors.SelectMany(n => n.SearchRequests).Where(s =>
!targetUser.SearchRequests.Any(ts => ts.Tag.ToLower() ==
s.Tag.ToLower())).Select(t=>t.Tag).Distinct();
            var scoredTags = new List<TagScore>();

            foreach (var tag in tags)
            {
                scoredTags.Add(new TagScore
                {
                    Name = tag,
                    Score = neighbors.Sum(n => n.Likes.Where(l => l.Post.Tags.Any(t =>
t.Name.ToLower() == tag.ToLower())).Count())
                });
            }

            return scoredTags.Where(t=>t.Score > 0).OrderByDescending(s =>
s.Score).Take(5).Select(s => new Tag
            {
                Name = s.Name
            }).ToArray();
        }

        private async Task<User[]> GetNearestNeighbors(User targetUser)
        {
            var users = await _dbContext.User.Take(100).Select(u => new User
            {
                Id = u.Id,
                SearchRequests = u.SearchRequests.OrderByDescending(s =>
s.Date).Take(10).ToList(),
                Likes = u.Likes.Select(l => new Like
                {
                    PostId = l.PostId,
                    UserId = l.UserId,
                    Post = new Post
                    {
                        Tags = l.Post.Tags
                    }
                }).ToList()
            }).ToArrayAsync();

            var neighbors = new List<NeighborUser>();
            foreach (var neighbor in users)
            {
                if (neighbor.Id == targetUser.Id) {
                    neighbors.Add(new NeighborUser
                    {
                        CosineSimilarity = double.NegativeInfinity
                    });
                    continue;
                }

                neighbors.Add(new NeighborUser
                {
                    UserId = neighbor.Id,
                    CosineSimilarity = CalculateCosineSimilarityBetweenUsers(targetUser,
neighbor)
                });

            }

            var nearestNeighbors =
neighbors.Where(n=>!double.IsInfinity(n.CosineSimilarity) &&
!double.IsNaN(n.CosineSimilarity)).OrderByDescending(n =>
n.CosineSimilarity).Take(10).ToArray();
```

```
            return users.Where(u => nearestNeighbors.Any(n => n.UserId ==
u.Id)).ToArray();
        }

        private double CalculateCosineSimilarityBetweenUsers(User targetUser, User
neighbor)
        {
            var topCoefficient =  0;
            var bottomACoefficient = 0;
            var bottomBCoefficient = 0;
            foreach (var tag in targetUser.SearchRequests)
            {
                var targetTagLikes = targetUser.Likes.Where(l => l.Post.Tags.Any(t =>
t.Name == tag.Tag)).Count() + 1;
                var neighborTagLikes = neighbor.SearchRequests.Any(s => s.Tag == tag.Tag)
?
                                            neighbor.Likes.Where(l => l.Post.Tags.Any(t =>
t.Name.ToLower() == tag.Tag.ToLower())).Count() + 1 : 0;
                topCoefficient += targetTagLikes * neighborTagLikes;
                bottomACoefficient += targetTagLikes * targetTagLikes;
                bottomBCoefficient  += neighborTagLikes *  neighborTagLikes;
            }

            return topCoefficient / (Math.Sqrt(bottomACoefficient) *
Math.Sqrt(bottomBCoefficient));
        }

    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
О.В. Ковтунець	*О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“13” квітня 2020 р.	“14” квітня 2020 р.

Інформаційна система підтримки сервісу для розповсюдження фотознімків

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-6221.1081-с.ТЗ*

на 11 сторінках

Київ – 2020 року

ЗМІСТ

					ДП ІС-6221.1081-с.ТЗ		
Зм.	Арк.	Прізвище	Підпис	Дата			
Розроб.		Севєрцев В.В.			Інформаційна система підтримки сервісу для розповсюдження фотознімків	Літ.	Лист 2 / Листів 11
Перевірив.		Ковтунець О.В.					
Н. кон.		Новінський В.П				КПІ ім. ІгоряСікорського кафедра АСОІУ гр. ІС-62	
Затв.		Павлов О.А.					

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи: «Інформаційна система підтримки сервісу розповсюдження фотознімків».

Коротке найменування системи: «Інформаційна система підтримки сервісу розповсюдження фотознімків».

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником проекту являється кафедра Автоматизованих систем обробки інформації та управління НТУУ "КПІ". Представником замовника є Ковтунець Олесь Володимирович. Адреса замовника: м. Київ, п. Перемоги 37, 18 корпус ФІОТ АСОІУ.

Розробником є студент групи ІС-62 кафедри Автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. І. Сікорського" Сєвєрцев Вадим Вячеславович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи.

Автоматизовані системи. Стадії створення;

- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при

створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий строк початку роботи по створенню системи 13 березня 2020 року. Плановий строк кінця роботи по створенню системи 31 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення розробки

Призначенням розробки є забезпечення механізмів збереження, пошуку, редагування та рекомендацій контенту для сервісу розповсюдження фотознімків.

2.1 Цілі створення системи

Цілями розробки є:

- Розробка сервісу розповсюдження фотознімків та інформаційної системи для його підтримки;
- забезпечення зручного способу обміну, розповсюдження та збереження зображень або фотознімків;
- забезпечення можливості оцінювати та шукати зображення та фотознімки за вподобаннями або потребами
- формування рекомендацій для користувачів сервісу відповідно до їх активності у сервісі.

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

1) Створити веб-сайт для сервісу;
2) Розробити інформаційну систему, що забезпечуватиме збір, обробку та доступ до даних сервісу:

- реєстрація користувачів;
- завантаження зображень та фотознімків до колекції сервісу;
- оцінювання зображень та фотознімків інших користувачів;
- пошук та фільтрація зображень;
- перегляд зображень з колекції сервісу.

3) Розробити та реалізувати алгоритм для рекомендації зображень користувачам, відповідно до їхньої активності у сервісі;

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об'єктом автоматизації є основні процесами функціонування сервісу для розповсюдження фотознімків, а саме: зберігання фотознімків та зображень, їх пошук, перегляд та розповсюдження між користувачами сервісу.

Розглянемо найбільш популярні аналогічні системи.

Pinterest

Pinterest - це медіа соціальна мережа, яка дозволяє користувачам ділитися зображеннями, пов'язаними з роботою, природою, товарами, відпочинком, послугами та іншими складовими нашого життя, та візуально відкривати нові інтереси, переглядаючи зображення, які розмістили інші користувачі ресурсу. Користувачі мають змогу підписуватися об'єднуватися у групи за вподобаннями та ділитися тематичними знімками один з одним.

Unsplash

Unsplash - це веб-сайт, призначений для обміну фотографіями за ліцензією Unsplash. Веб-сайт заявляє про понад 110 000 фотографів, що надсилають участь, і створює понад 11 мільярдів переглядів фотографій на місяць у своїй зростаючій бібліотеці фотографій.

Unsplash названий одним із провідних веб-сайтів у галузі фотографії.

Фотографи завантажують фотографії на веб-сайт, а команда редакторів Unsplash займається їх подальшим просуванням. Дозвільні умови авторських прав на його фотографії призвели до того, що Unsplash став одним з найбільших постачальників фотографій в Інтернеті, а фотографії користувачів часто з'являються у статтях.

We heart it

We Heart It – це візуальна платформа, яка підтримує зображення, анімовані GIF-файли та відео. Користувачі можуть зберігати (або «сердечно позначати») свої улюблені зображення, щоб поділитися ними з друзями та організувати їх у колекції.

Наведені вище системи дають користувачу змогу обрати теми, які їм цікаві. Далі ці теми використовуються системами для наповнення їх стрічки контенту зображеннями з заданою тематикою.

Розроблювана ж система дає користувачу змогу виконувати пошук зображень за тегами та буде наповнювати стрічку контенту зображеннями з шуканою тематикою, але також, в залежності від активності користувача, система буде шукати ті зображення, що можуть сподобатися користувачу, але пошук яких він не виконував, та додавати їх у стрічку контенту користувача.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Розроблений програмний продукт повинен виконувати наступні функції:

- функція завантаження зображень та фотознімків користувачів до колекції системи
- функція формування каталогу зображень для користувачів
- функція формування рекомендацій для користувачів
- функція пошуку зображень серед колекції системи
- функція видалення власних зображень користувачів з колекції системи

4.2 Вимоги до надійності

Система повинна зберігати працездатність та забезпечувати відновлення своїх функцій при виникненні наступних ситуацій:

- при не постійному електропостачання апаратної частини в системі, що призводять до перезавантаження ОС, відновлення повинно відбуватись після перезапуску ОС і запуску виконуючого файлу системи;

- при помилках, зв'язаних з програмним забезпеченням (ОС і драйвери пристроїв), відновлення працездатності покладається на ОС;

- дані облікових записів користувачів повинна бути конфіденційною та захищатися від не санкціонованого доступу.

4.3 Умови експлуатації

Для нормальної експлуатації розроблюваної системи має бути забезпечене безперебійне живлення персональної електронно-обчислювальної машини.

Періодичне технічне обслуговування і тестування технічних засобів повинні включати в себе обслуговування і тестування всіх використовуваних

засобів, включаючи робочі станції, сервери, кабельні системи та мережеве обладнання, пристрої безперебійного живлення.

Розміщення обладнання, технічних засобів повинно відповідати вимогам техніки безпеки. Всі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

4.4 Вимоги до складу і параметрів технічних засобів

Вимоги до серверу

Для роботи системи необхідні наступні параметри системи:

– Процесор с частотою 1,6 ГГц або потужніший.

– ОЗУ об’ємом 1.5 ГБ

– 10 ГБ доступного простору на жорсткому диску або на SSD

– Жорсткий диск с частотою обертання 5 400 об/хв або еквівалентний йому SSD

– Відеокарта с підтримкою DirectX 11 і розширення екрану 1024x768 або вище

Вимоги до клієнта

Наявність одного з веб-браузерів:

- Opera 9.6+;
- Internet Explorer 9.0+;
- Chrome 6.0+;

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ етапу	Назва етапів виконання дипломного проекту	Строк виконання
1.	*Вивчення рекомендованої літератури*	*13.04.2020*
2.	*Аналіз існуючих методів розв'язання задачі*	*19.04.2020*
3.	*Постановка та формалізація задачі*	*22.04.2020*
4.	*Розробка інформаційного забезпечення*	*24.04.2020*
5.	*Алгоритмізація задачі*	*25.04.2020*
6.	*Обґрунтування використовуваних технічних засобів*	*29.04.2020*
7.	*Розробка програмного забезпечення*	*12.05.2020*
8.	*Налагодження програми*	*25.05.2020*
9.	*Виконання графічних документів*	*20.05.2020*
10.	*Оформлення пояснювальної записки*	*21.05.2020*
11.	*Подання ДП на попередній захист*	*13.05.2020*
12.	*Подання ДП на основний захист*	*06.06.2020*
13.	*Подання ДП рецензенту*	*08.06.2020*

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Види випробувань узгоджуються з замовником до проведення випробувань. Здача - прийом робіт виконується поетапно у відповідності з робочою програмою та календарним планом.

Всі програмні продукти, що створюються в рамках даної системи передаються замовнику як у вигляді готових модулів, так і у вигляді вихідних кодів, представлених в електронній формі.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003919975

Дата перевірки:
10.06.2020 02:25:32 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2020 14:39:14 EEST

ID користувача:
77149

Назва документу: Severcev_bachelor_is62

ID файлу: 1003935272 Кількість сторінок: 60 Кількість слів: 6599 Кількість символів: 51056 Розмір файлу: 7.15 MB

10.4% Схожість

Найбільша схожість: 3.55% з джерело бібліотеки. ID файлу: 1000018093

4.61% Схожість з Інтернет джерелами	40	Page 62
10.2% Текстові збіги по Бібліотеці акаунту	223	Page 62

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

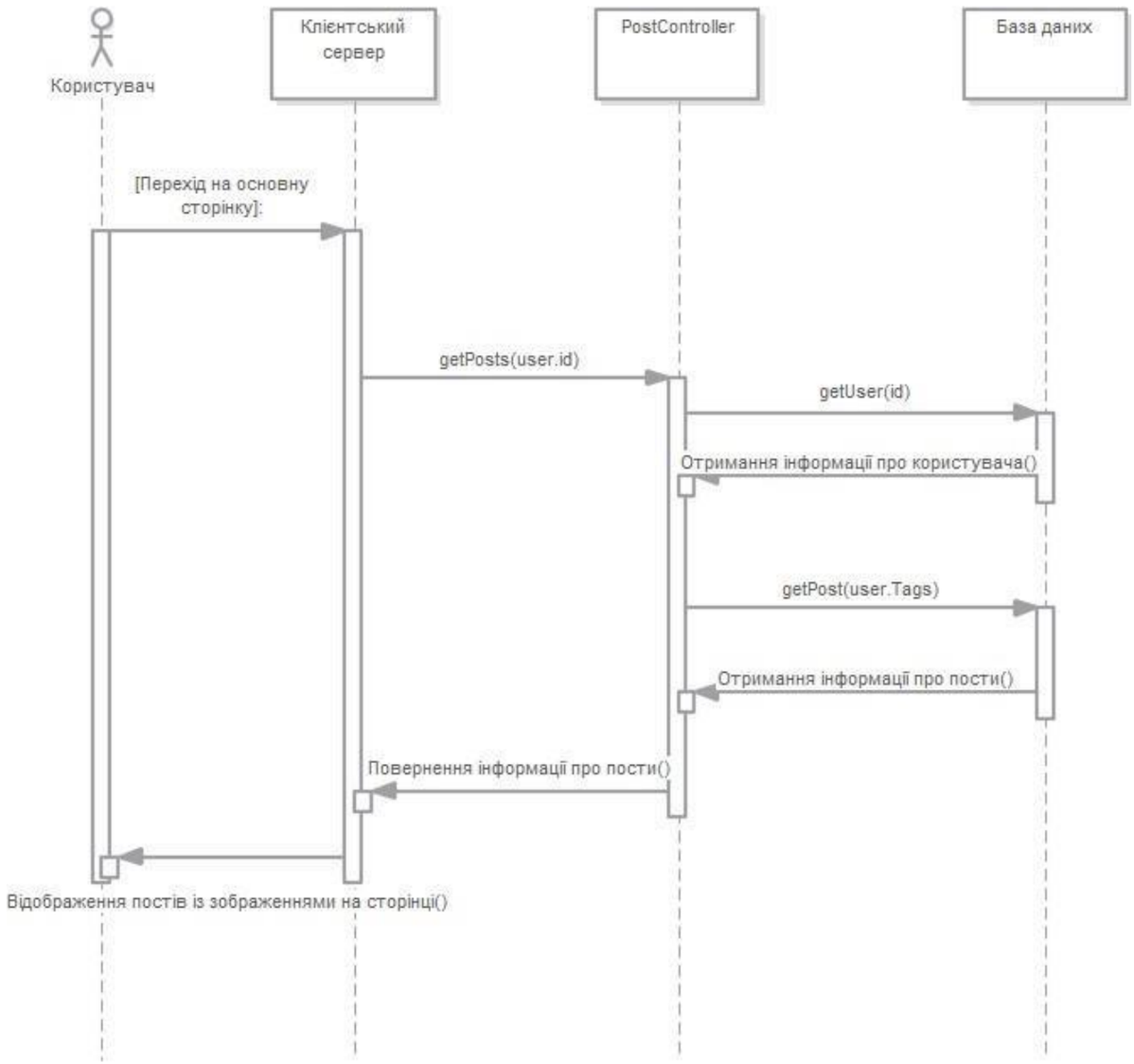
Підміна символів

Заміна символів	8

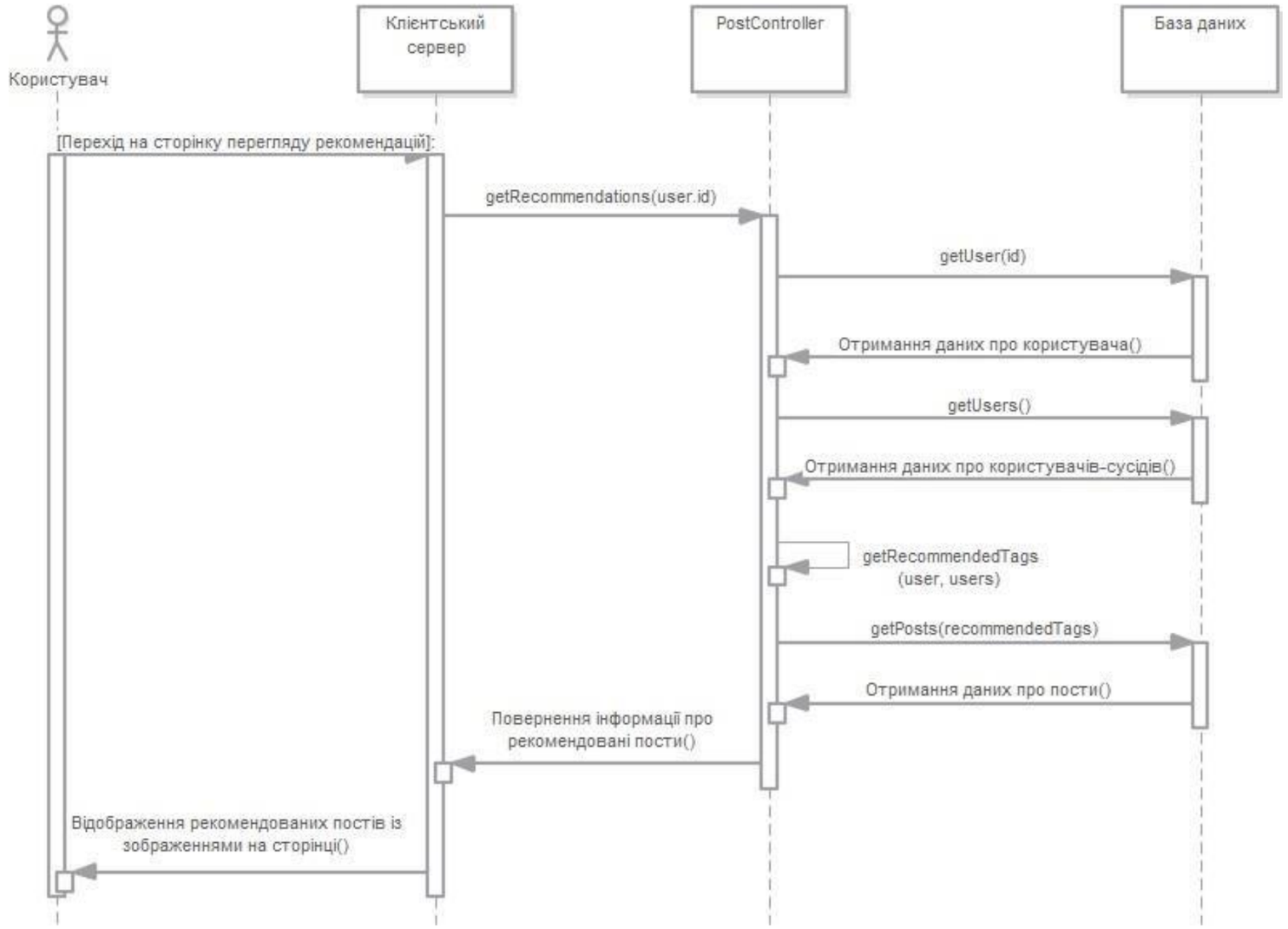
Графічний матеріал
до дипломного проєкту

на тему: «Інформаційна система підтримки сервісу розповсюдження фотознімків»

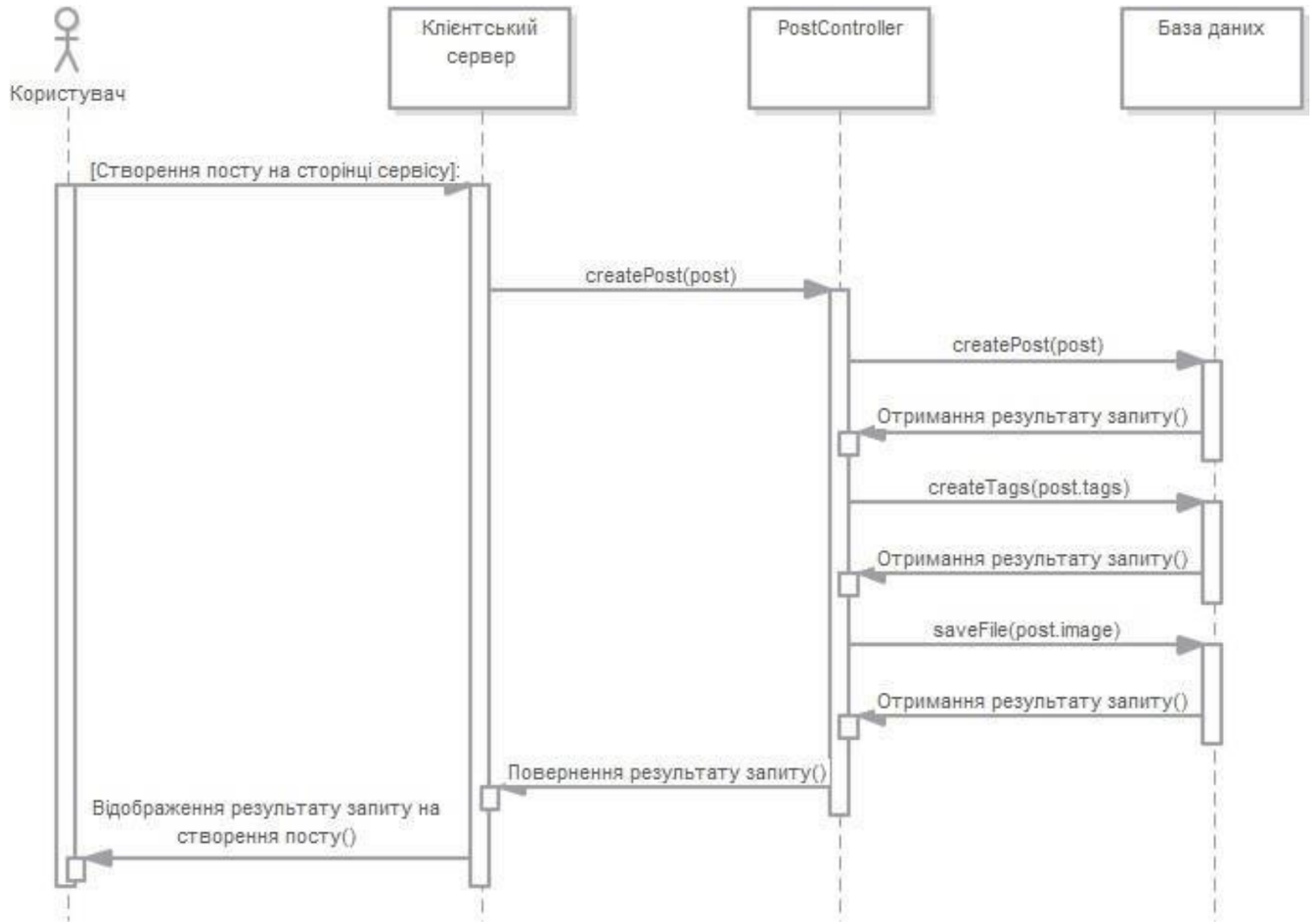
Київ – 2020 року



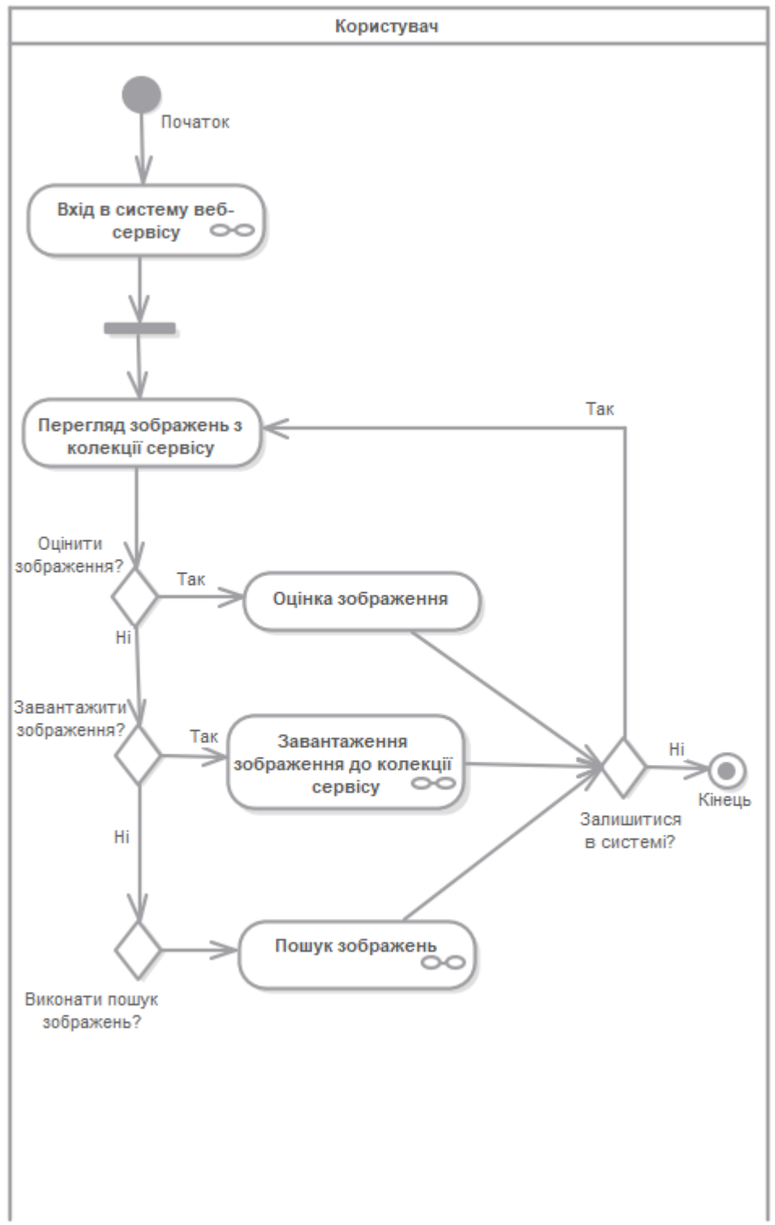
					ДП ІС-6221.1081-с.ССП		
						Літера	Маса	Масштаб
Зм.	Арк.	документа	Підпис	Дата	Схема структурна діаграми послідовності процесу завантаження зображень на основній сторінці			
Розробив		Сєвєрцев В.В.				Аркуш 1	Аркушів 1	
Перевірив		Ковтунець О.В.						
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						



					ДП ІС-6221.1081-с.ССП		
Зм.	Арк.	документа	Підпис	Дата	Схема структурна діаграми послідовності процесу завантаження рекомендацій	Літера	Маса / Масштаб
Розробив		Сєвєрцев В.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					

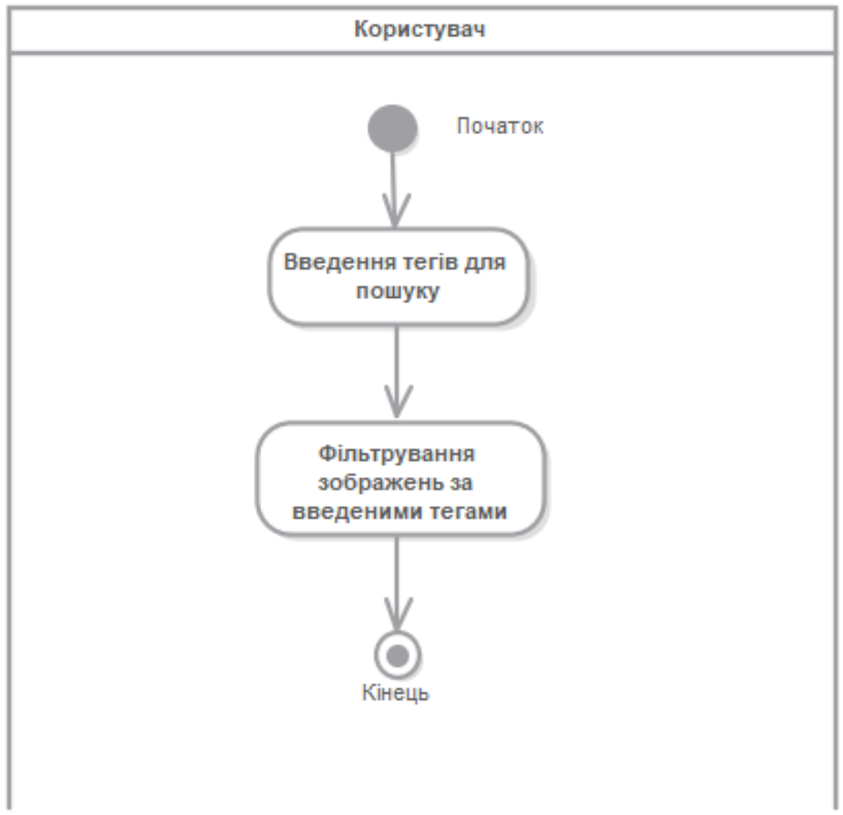


					ДП ІС-6221.1081-с.ССП		
					Схема структурна діаграми послідовності процесу завантаження зображення користувача до колекції сервісу	Літера	Маса / Масштаб
Зм.	Арк.	документа	Підпис	Дата			
Розробив		Сєвєрцев В.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					



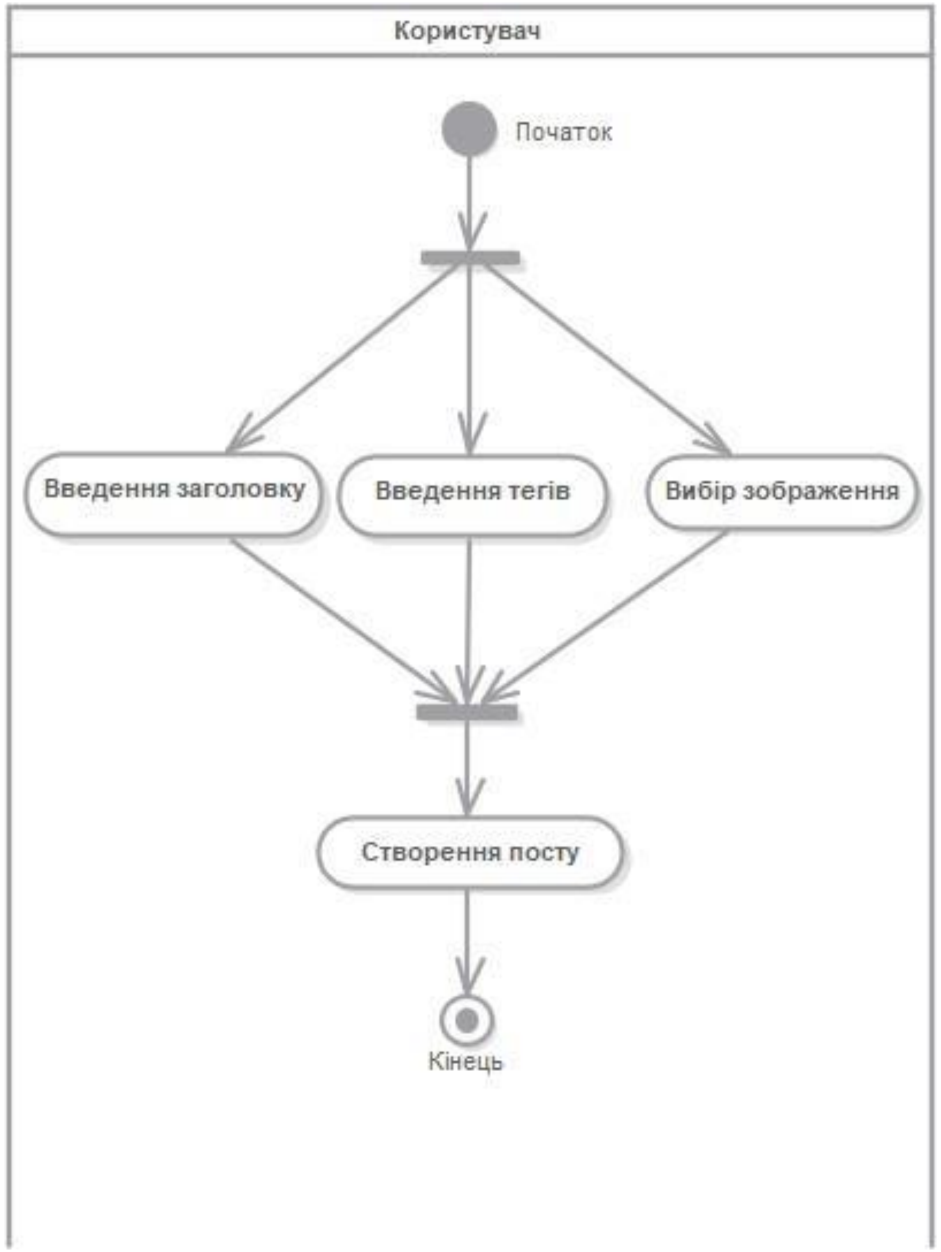
					ДП ІС-6221.1081-с.ССД		
					Схема структурна діаграми діяльності загальна	Літера	Маса / Масштаб
Зм.	Арк.	документа	Підпис	Дата			
Розробив		Сєвєрцев В.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					

					ДП ІС-6221.1081-с.ССД		
					Схема структурна діаграми діяльності авторизації користувача в системі	Літера	Маса / Масштаб
Зм.	Арк.	документа	Підпис	Дата			
Розробив		Сєвєрцев В.В.					
						Аркуш 1	
Перевірив		Ковтунець О.В.					
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					



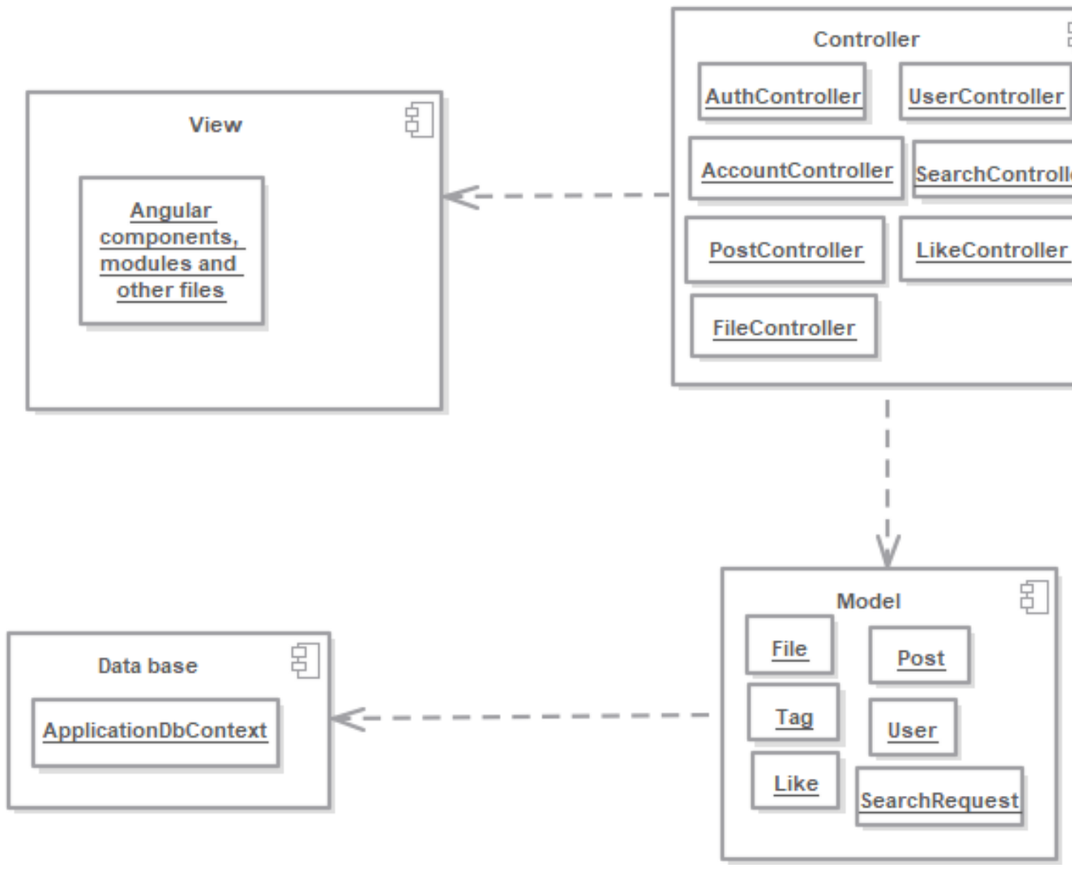
					ДП ІС-6221.1081-с.ССД			
Зм.	Арк.	документа	Підпис	Дата	Схема структурна діаграми діяльності пошуку зображень	Літера	Маса	Масштаб
Розробив		Сєвєрцев В.В.						
						Аркуш 1		
Перевірив		Ковтунець О.В.						
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						

ДП ІС-6221.1081-с.ССД



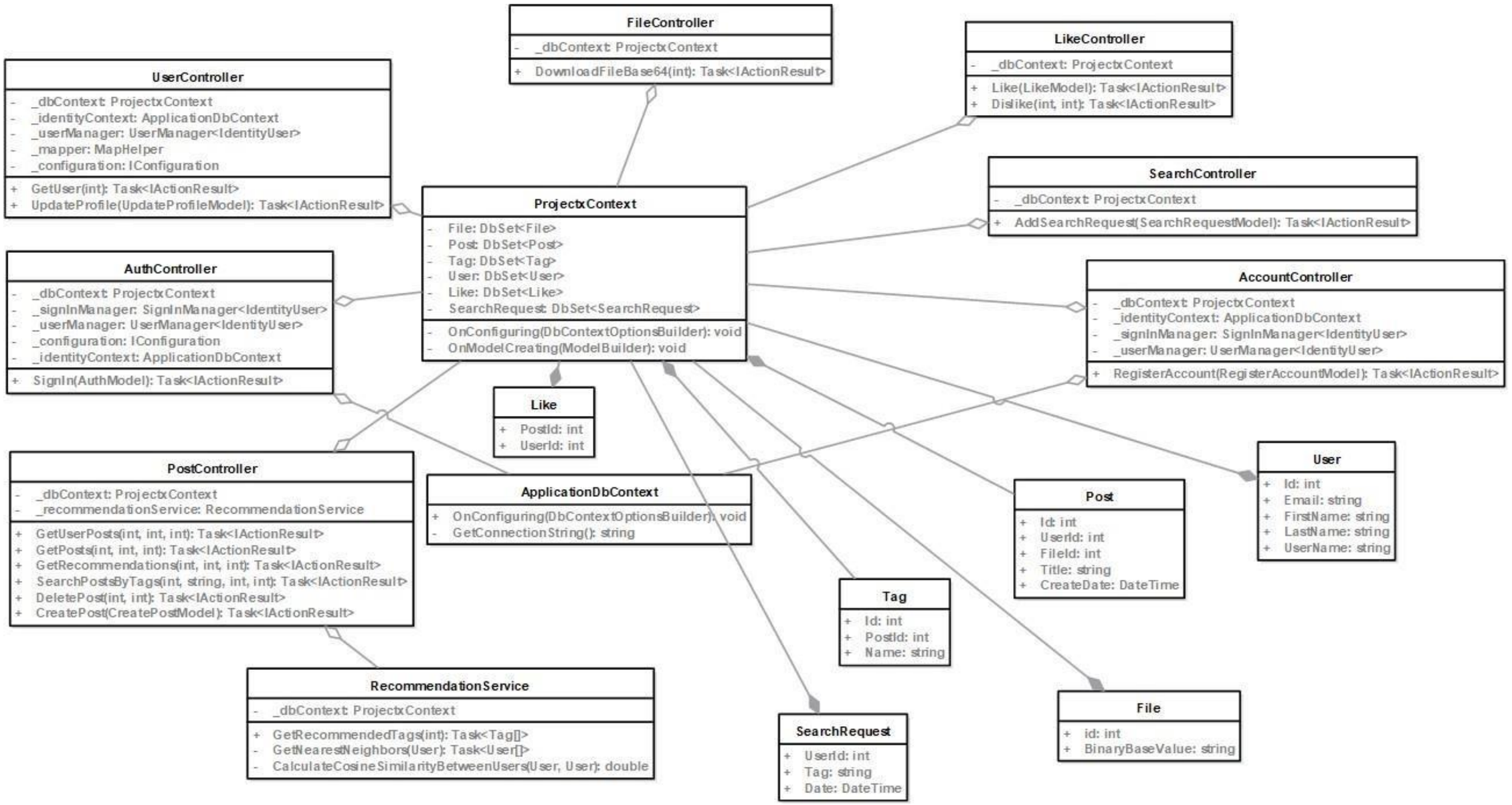
Зм.	Арк.	документа	Підпис	Дата	ДП ІС-6221.1081-с.ССД
Розробив	Сєвєрцев В.В.				Схема структурна діаграми діяльності створення посту
Перевірив	Ковтунець О.В.				Літера / Маса / Масштаб / Аркуш 1
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків
Н. кон.	Новінський В.П.				КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62
Затвердив	Ковтунець О.В.				

ДП ІС-6221.1081-с.ССК



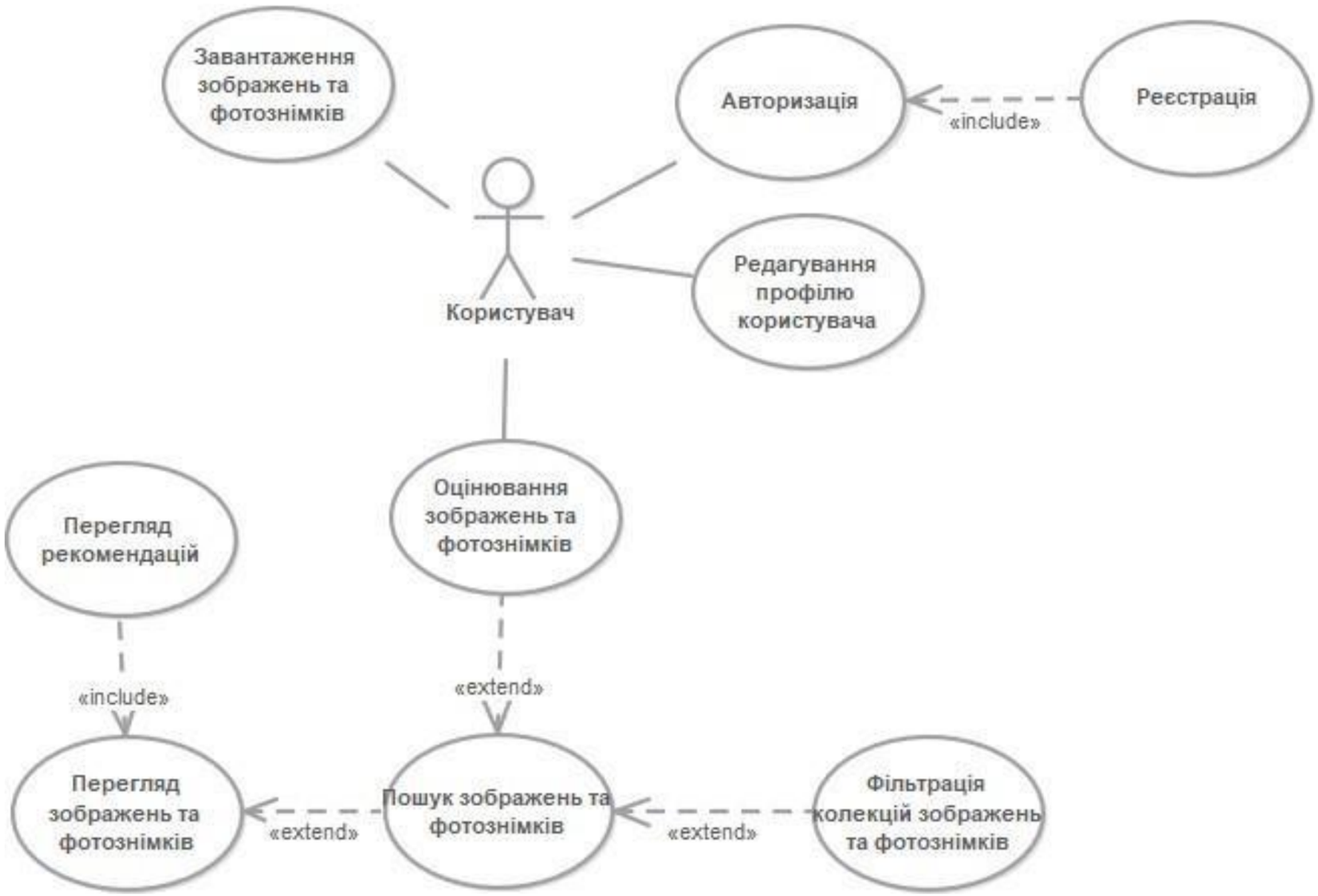
					ДП ІС-6221.1081-с.ССК		
						Літера	Маса / Масштаб
Зм.	Арк.	документа	Підпис	Дата	Схема структурна діаграми компонентів системи		
Розробив		Сєвєрцев В.В.				Аркуш 1	
Перевірив		Ковтунець О.В.					
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					

ДП ІС-6221.1081-с.ССК



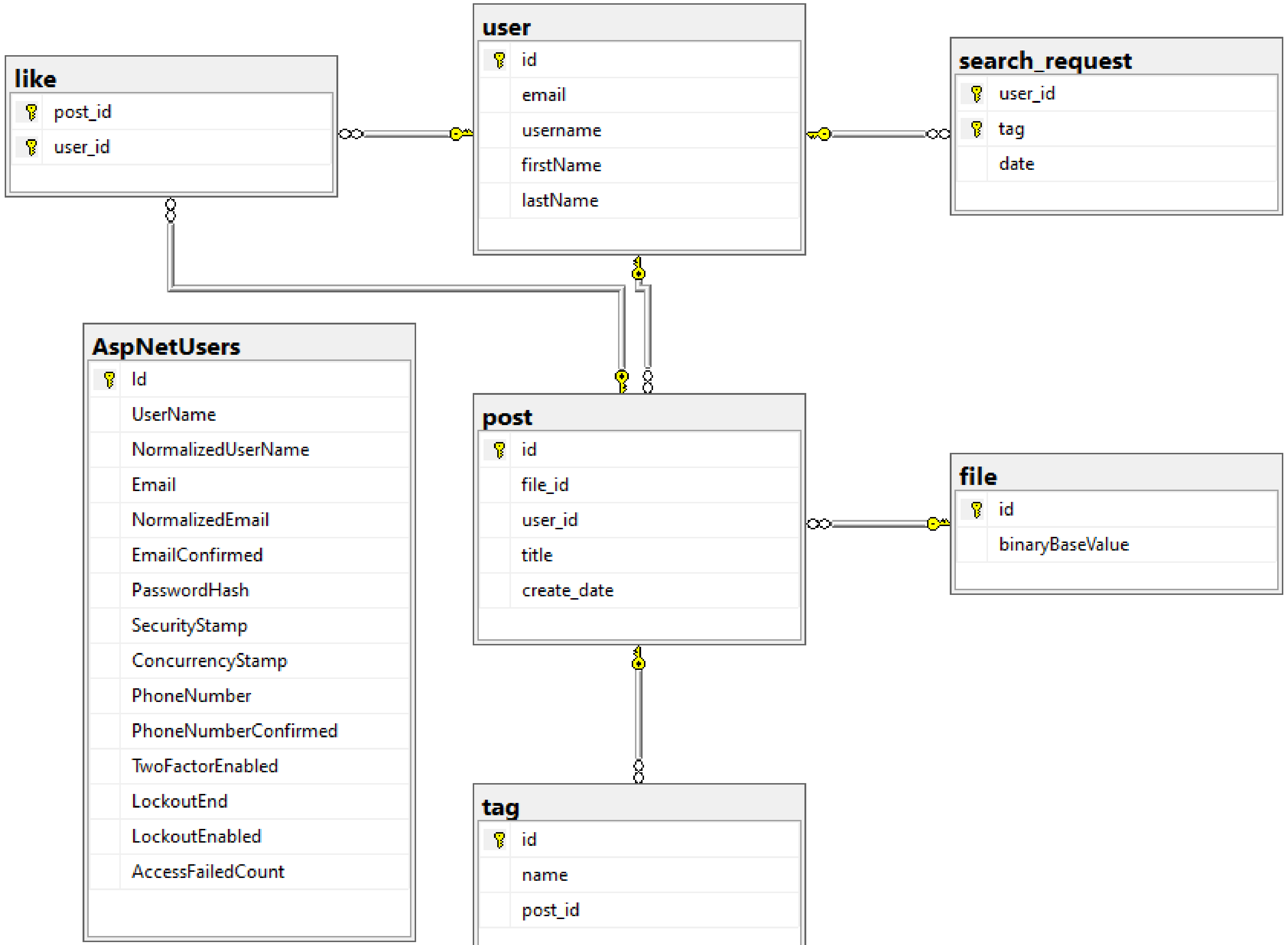
					ДП ІС-6221.1081-с.ССК			
						Літера	Маса	Масштаб
Зм.	Арк.	документа	Підпис	Дата	Схема структурна діаграми класів			
Розробив		Сєвєрцев В.В.						
						Аркуш 1		
Перевірив		Ковтунець О.В.						
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						

ДП ІС-6221.1081-с.ССВ



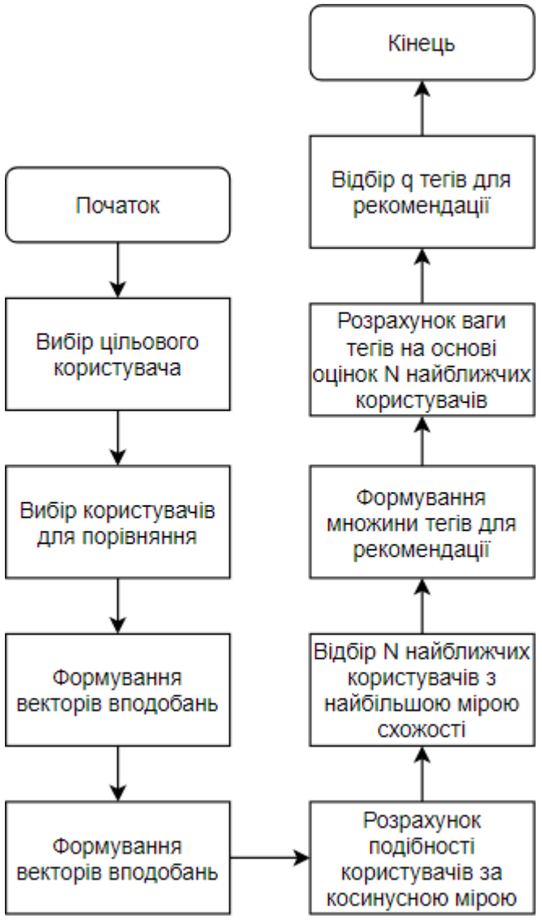
					ДП ІС-6221.1081-с.ССВ		
					Схема структурна варіантів використання	Літера	Маса / Масштаб
Зм.	Арк.	документа	Підпис	Дата			
Розробив		Сєвєрцев В.В.					
Перевірив		Ковтунець О.В.				Аркуш 1	
Т. кон.					Інформаційна система підтримки сервісу розповсюдження фотознімків	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62	
Н. кон.		Новінський В.П.					
Затвердив		Ковтунець О.В.					

ДП ІС-6221.1081-с.СБД



					ДП ІС-6221.1081-с.СБД		
Зм.	Арк.	документа	Підпис	Дата	*Схема бази даних*	Літера	Маса	Масштаб
Розробив		Сєверцев В.В.						
Перевірив		Ковтунець О.В.				Аркуш 1		
Т. кон.					*Інформаційна система підтримки сервісу розповсюдження фотознімків*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-62*		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						

Рішення з математичного забезпечення



Постановка задачі

Існує множина користувачів сервісу. Кожен користувач має власну множину пошукових запитів серед зображень із колекції сервісу. Також існує множина постів із зображеннями. Кожен пост має власну множину тегів , що задаються користувачами при створені постів. В ході використання сервісу користувачі виконують пошук зображень за тегами та відмічають ті зображення, які їм сподобалися.

Користувач в більшій мірі буде виконувати пошук та переглядати ті зображення, які йому більш цікаві, ніж інші. Також, користувачі, що шукають та переглядають пости за схожими тегами можуть мати спільні вподобання.

Для формування рекомендацій необхідно оцінювати активність користувачів при користуванні сервісом та, при необхідності, створювати рекомендації для користувачів в залежності від їх вподобань. Дану задачу можна вирішити за допомогою методу колаборативної фільтрації.

Метод розв'язання

Колаборативна фільтрація на основі схожості вподобань користувачів з відбором k-найближчих сусідів.

Висновки

Для вирішення задачі було обрано метод колаборативної фільтрації з відбором k-наближчих сусідів. Для розрахунку міри схожості користувачів використовуватиметься косинусна міра.

Демонстраційний плакат до дипломного проекту

«Інформаційна система підтримки сервісу розповсюдження фотознімків»

Виконав студент гр. ІС-62 Сєвєрецев В.В

Керівник ДП Ковтунець О.В.